UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

Automatisk deteksjon av menneskeskapte objekter med optisk kamera fra autonome undervannsfarkoster

Ragnar Smestad
Masteroppgave høsten 2014

Automatisk deteksjon av menneskeskapte objekter med optisk kamera fra autonome undervannsfarkoster

Ragnar Smestad

21. juli 2014

Sammendrag

Bakgrunn og motivasjon Autonome undervannsfarkoster (AUV-er) er ubemannede farkoster som uten kontinuerlig, menneskelig fjernstyring utfører oppdrag under vann. Det er ønskelig å gjøre dem adaptive, slik at de har evne til å tilpasse sine handlinger underveis i et tokt ut fra egne vurderinger. Slike beslutninger tas på grunnlag av innsamlet sensorinformasjon. Sensordata som brukes i denne oppgaven er optiske bilder av havbunnen.

Motivasjonen er at farkosten skal kunne endre sin oppførsel basert på innholdet i bildene, for eksempel dersom det tas bilde av en bestemt type objekter. Mer konkret er det ønskelig å skille menneskeskapte objekter fra naturlige objekter og formasjoner på havbunnen.

Deteksjon av objekter i undervannsbilder er også interessant i forbindelse med filtrering av innsamlet datamateriale i etterkant av et tokt. Å automatisk kunne skille ut bilder av potensiell interesse vil kunne gi store tids- og kostnadsbesparelser sammenlignet med manuell inspeksjon.

Metode I denne masteroppgaven presenteres en algoritme for automatisk deteksjon av menneskeskapte objekter i optiske undervannsbilder. «Menneskeskapt» defineres som et objekt der konturene eller andre deler av objektet inneholder rette linjer. Videre kreves det at minst to linjer er enten parallelle eller står normalt på hverandre.

Preprosessering består av geometrisk korreksjon og utjevning av belysning med homomorf filtrering. Deretter benyttes fasekongruens til segmentering av kanter og linjer. Til deteksjon av segmenterte linjer brukes radontransform. Videre benyttes en egenutviklet fremgangsmåte for gruppering av detekterte kanter til sammensatte deteksjoner: to og to linjesegmenter sammenlignes for å lokalisere parallelle eller ortogonale kanter. Slike hjørner og paralleller er grunnkomponenter i objektdeteksjonen, og omtales som «primitiver». Dersom to eller flere primitiver deler en kant, vil disse slås sammen til en større gruppe som representerer én deteksjon. Det utvikles et mål på tiltro til at en deteksjon svarer til et menneskeskapt objekt, som øker for hvert nye primitiv som knyttes til en deteksjon.

Resultater og konklusjon Ved testing av algoritmen på et større testsett oppnås en TPR (*True Positive Rate*) på 0.75 som beste resultat. For denne verdien er laveste mulige FPR (*False Positive Rate*) 0.75. FPR kan reduseres betraktelig ved å stille strengere krav til deteksjonskvalitet (tiltro), men dette går relativt mye på bekostning av antall korrekte deteksjoner. FPR er regnet som antall feildeteksjoner per bilde.

Innhold

Forord

Denne rapporten dokumenterer forfatterens masteroppgave (M.Sc.) som har vært en del av *Profesjonsstudiet i informatikk, Robotikk og intelligente systemer*, ved institutt for informatikk, Universitetet i Oslo. Oppgaven er skrevet for Forsvarets Forskningsinstitutt (FFI) på Kjeller.

Først og fremst vil jeg rette en stor takk til mine veiledere, Øivind Midtgaard (FFI), Anne H. S. Solberg (IFI) og Jim Tørresen (IFI) for uvurderlig hjelp gjennom hele prosessen. Jeg har satt stor pris på dette. Arbeidet har gitt meg faglig kunnskap og gjort meg mange erfaringer rikere.

Takk til Trond Heldal Hagen for mat, pauser og skravling på Kjeller. Tidlige frokoster, sene middager og mang en kaffepause er noe jeg ser tilbake på som gode minner.

Sist men ikke minst vil jeg takke Siri for at du har holdt ut med meg i denne hektiske perioden, selv om jeg ikke har vært mye til selskap for deg. Tenk at du likevel sa *ja*, midt oppe i det hele! Du er fantastisk!

Kapittel 1

Innledning

Hverken undervannsfarkoster eller undervannsavbildning er helt nye fenomener, men akkurat hvor og når mennesket for første gang begynte lage bilder under vann er vanskeligere å fastslå [22]. Det som i midlertid er sikkert, er at fra da David Bushnell sjøsatte sin miniubåt «Turtle» i 1775 [6] og William Thompson senket sitt kamera i sjøen og tok historiens første undervannsfotografi i 1856 [1], har det vært en voldsom utvikling til dagens selvstyrte, «intelligente» farkoster og høyteknologiske digitalkameraer.

Undervannsfarkoster kan deles inn i to hovedgrupper: bemannede- og ubemannede systemer. De bemannede er kjent for de fleste, der det finnes alt fra store militære ubåter til mindre, sivile varianter som benyttes til forskning og turisme. Ubemannede farkoster deles videre inn flere undergrupper. Det finnes relativt enkle typer som påmonteres ulike sensorer og passivt taues av skip, og mer avanserte farkoster som fjernstyres ved hjelp av kabel[1] eller akustiske signaler[2]. En siste undergruppe, autonome undervannsfarkoster (AUV)[3], er selvstendige systemer som bærer med seg sin egen energikilde og utfører forhåndsbestemte oppgaver uten videre behov for kommunikasjon mens oppdraget pågår [6]. Nettopp fordi de utfører komplekse arbeidsoppgaver autonomt eller «automatisk», omtales de også gjerne som undervanns*roboter*[4]. Autonome undervannsfarkoster er særlig egnet til å samle inn sensordata for kartlegging og dokumentasjon av havbunnen på store dyp.

Forsvarets forskningsinstitutt (FFI) har, sammen med den norske industribedriften Kongsberg Maritime (KM), utviklet HUGIN AUV. Denne AUV-en er internasjonalt markedsledende, og kan brukes innen anvendelser som offshore olje og gass, havforskning, miljøkartlegging og søk etter objekter. En state-of-the-art HUGIN AUV er utrustet med tre forskjellige avbildende sensorer: interferometrisk syntetisk apertur sonar (SAS), høyoppløselig multistråle ekkolodd (MBE) og optisk kamera. Alle disse sensore-

[1] Remotely Operated Vehicle (ROV)
[2] Untethered Underwater Vehicle (UUV)
[3] Autonomous Underwater Vehicle (AUV)
[4] *Robot: a machine capable of carrying out a complex series of actions automatically, especially one programmable by a computer* [11].

ne produserer bilder av havbunnen. AUV-operasjoner som omfatter søk etter spesifikke objekter baserer seg på at man først avbilder et større område med interferometrisk SAS. Så lages det en liste av mulige objekter man ønsker å studere nøyere (automatisk eller manuelt). Deretter lages en toktplan (automatisk eller manuelt) som bringer AUV-en nær nok objektene til å ta kamerabilde av objektene.

For en autonom farkost er det et mål at den selv kan endre sin oppførsel ut fra hva det er som har blitt tatt bilde av. Mulige handlinger kan være å gjøre nye passeringer over et objekt for å sikre at hele objektet avbildes eller å få kartlagt et område rundt funnet. Det kan også være aktuelt å gå ned i høyde for bedre detaljer, eller for eksempel følge et rør. En autonom gjennomføring er da avhengig av at roboten har kunnskap om relevante objekter som kan avbildes, slik at riktig aksjon kan gjennomføres. For å få til dette vil det være behov for en algoritme som detekterer og klassifiserer objektene i kamerabildene.

En annen fordel med automatisk deteksjon av objekter i kamerabilder er at en menneskelig operatør kan slippe å inspisere hvert eneste bilde fra et helt tokt på leting etter avbildede objekter. Med en bilderate på opptil 4 bilder per sekund og med tokt på inntil 20 timer vil det være både tidkrevende, dyrt og ikke minst kjedelig å se gjennom 288,000 bilder av mudderbunn!

1.1 Mål for oppgaven

Målet for dette masterprosjektet er å gjøre automatisk deteksjon av menneskeskapte objekter i optiske undervannsbilder samlet inn av HUGIN AUV.

Arbeidet deles opp i følgende to hovedpunkter:

- Utvikle en detektor som kan skille mellom menneskeskapte og andre objekter
- Evaluere detektoren på et større datasett

Menneskeskapte objekter Bildene som er plukket ut til bruk i denne oppgaven er hentet fra et datasettet av begrenset størrelse, men med god variasjon blant bilder både med og uten objekter. Utvalget er gjort uten å ta hensyn til noen modell for menneskeskapte objekter, og avbildede objekter omfatter alt fra vrakgods til rørledninger, søppel, betongelementer, øvingsminer og oljefat. For å begrense omfanget av oppgaven er definisjonen av «menneskeskapt» konkretisert som følger:

> *Et menneskeskapt objekt er et objekt der konturene eller andre deler av objektet inneholder rette linjer. Videre kreves det at minst to linjer er enten parallelle eller står normalt på hverandre.*

Definisjonen tar utgangspunkt i at menneskeskapte objekter gjerne er utformet maskinelt og består av enkle geometriske former som rektangler, sirkler, trekanter og lignende. I denne omgang er det valgt å bare fokusere på rektangler. Med denne

definisjonen er det åpenbart at det er vanskelig å detektere *alt* som er menneskeskapt. Blant de forskjellige objektene som ble nevnt vil det være flere som ikke passer med modellen. Perspektiv, eller hvor et avbildet objekt ligger i bildeflaten, spiller også inn. For eksempel vil tverrbåndene til et oljefat se buede ut fra siden, men se rette ut når oljefatet ligger rett under kameraet.

For å kunne gjøre gode deteksjoner av rette linjer, blir det utført geometrisk korreksjon på bildene. Dette veier opp for eventuell linseforvrengning, slik at linjer som er rette i virkeligheten også ser rette ut i bildet. Kalibrering av kameraet og implementering av geometrisk korreksjon er ikke en del av oppgaven.

1.2 Disposisjon

Disposisjonen i denne oppgaven vil avvike noe fra den typiske IMRAD-modellen[5]. Arbeidet kan karakteriseres som en evolverende prosess, der utfallet av eksperimentene i den første fasen er avgjørende for hvilke metoder som velges for veien videre. På grunn av dette vil det være naturlig å analysere virkningen av de enkelte metodene underveis, og i den endelige resultatdelen presenteres i stedet bare ytelsen til detektoren som utvikles. Det kan allerede nå nevnes at siden dokumentets volum stadig har økt i omfang, har det blitt besluttet å eksemplifisere delresultater på bare et fåtall av bilder. De ulike delene av oppgaven er beskrevet i egne kapitler, som det gis en kort beskrivelse av under.

Kapittel 2: ***HUGIN AUV*** gir en introduksjon til undervannsfarkoster og ulike former for avbildning under vann. Det tas utgangspunkt i farkosten som har samlet inn bildene som brukes i oppgaven.

Kapittel 3: ***Introduksjon til prosessering av undervannsbilder*** tar for seg temaet optisk undervannsavbildning og utfordringer knyttet til dette.

Kapittel 4: ***Inspeksjon av havbunnsbilder*** viser et utvalg av bildene som er brukt i oppgaven og det blir kort diskutert hva som karakteriserer dem og eventuelle objekter på havbunnen.

Kapittel 5: ***Plan for oppgaven*** tar utgangspunkt i karakteristikken fra kapittel 4, og det gjøres et valg for hvilke metoder som skal testes i det videre arbeidet.

Kapittel 6: ***Forprosessering*** beskriver valgte metoder fra kapittel 5 som gjelder forprosessering av undervannsbildene.

Kapittel 7: ***Segmentering av objekter på sjøbunnen*** handler om resten av metodene som ble valgt i kapittel 5, som er ulike tilnærminger til deteksjon av objekter. De to hovedretningene som testes er kantdeteksjon og deteksjon basert på segmentering av skygger. I slutten av kapittelet diskuteres metodene som har blitt testet, og metoder for utviklingen av en detektor velges.

[5] Introduction, Method, Results and Discussion

Kapittel 8: ***Utvikling av detektoren*** dokumenterer arbeidet med å utvikle programvare for deteksjon av menneskeskapte objekter. I slutten av kapittelet gjøres det noen eksperimenter på treningssettet for å komme fram til fornuftige parameterinnstillinger for detektoren, før en endelig test på testsettet. I dette kapittelet beskrives også verktøy for vurdering av detektorens ytelse.

Kapittel 9: ***Resultater*** presenterer resultatene etter test av detektoren. Resultatene vises som en ROC-kurve (defineres i kapittel 8), i tillegg til noen eksempelbilder.

Kapittel 10: ***Diskusjon og konklusjon*** inneholder en generell diskusjon av implementasjonen, og oppsummerer arbeidet med oppgaven. Til slutt gis forslag til videre arbeid.

Programkode Programkoden til den utviklede detektoren vil være tilgjengelig på http://ragnarsm.at.ifi.uio.no/masteroppgave/.

Kapittel 2

HUGIN AUV

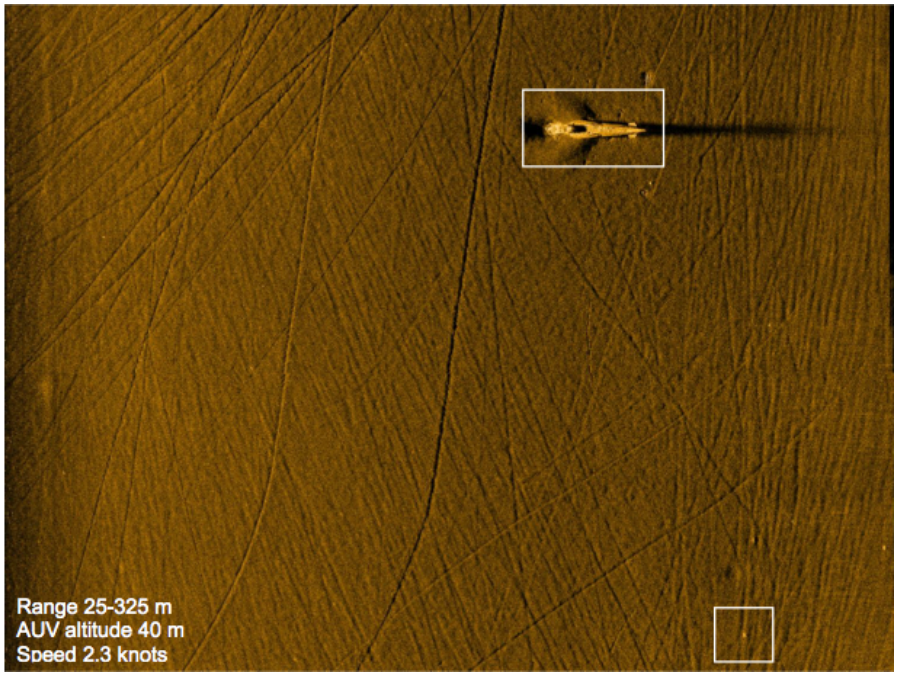
I dette kapittelet beskrives robotplatformen som samler inn bildene og noen av dens sensorer. Det meste av informasjonen i dette kapittelet er hentet fra [33].

2.1 AUV

Autonome undervannsroboter, eller *AUV-er*, er ubemannede farkoster som uten kontinuerlig, menneskelig fjernstyring utfører oppdrag under vann. Robotene finnes i ulike fasonger og størrelser, fra mindre enn 50 kg til flere tonn. Typiske anvendelser har vært innen offshore olje- og gassindustri, som for eksempel kartlegging og avbildning av havbunnen, inspeksjon av strekninger med rør og kabler, eller inspeksjon av objekter på havbunnen. Undervannsfarkoster er også populære i andre miljøer som havforskning, fiskeriforskning og miljøovervåkning. Etter hvert har AUV-er også fått flere militære anvendelsesområder, blant annet å finne miner og ellers utføre annen informasjonsinnhenting og overvåkning.

Fordelene med ubemannede fartøy er åpenbart en redusert risiko for tap av menneskeliv under operasjoner i farlige omgivelser, men også at robotene kan arbeide raskt og kostnadseffektivt. Ved å overlate automatiserbare oppgaver til AUV-er får man frigjort menneskelig kapasitet, som er allsidig men dyr. Farkostene er i stand til å operere på svært store dyp og kan kontinuerlig følge havbunnen i en gitt høyde. Dette gjør det mulig å samle inn sensordata som er mye bedre enn hva man kan få til fra overflategående fartøy. I militær sammenheng er det også en fordel at AUV-en kan være vanskeligere å oppdage og kan operere skjult bak fiendens linjer.

At farkostene er autonome, betyr at de er i stand til å utføre et forhåndsprogrammert tokt uten menneskelig fjernstyring. Videre er det ønskelig å også gjøre dem adaptive, slik at de i tillegg har en evne til å tilpasse sine handlinger underveis i oppdraget ut fra egne vurderinger. Slike beslutninger tas på grunnlag av innsamlet sensorinformasjon, for eksempel at AUV-en ikke avslutter toktet før et forhåndsdefinert område er fullstendig dekket av sonarbilder, eller at den navigerer på egenhånd langs en rørledning som skal inspiseres.



Figur 2.1: *Eksempel på bilde laget ved hjelp av HISAS 1030. Illustrasjon fra [33].*

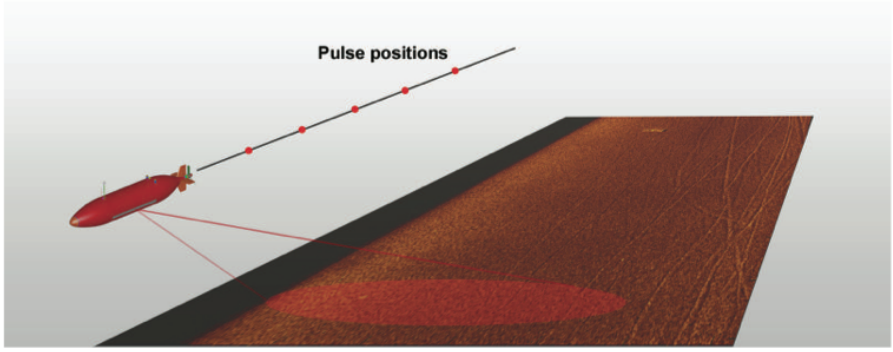
Utviklingen av HUGIN AUV ble startet tidlig på 1990-tallet som et samarbeidsprosjekt mellom Statoil, Forsvarets forskningsinstitutt (FFI), Norsk Undervannsintervensjon (NUI) og Kongsberg Maritime (KM). Farkosten kommer i flere forskjellige utgaver, med lengder fra 4,5 – 6 meter og en vekt på omkring 700 kg i luft. Alle HUGIN-modeller er vektnøytrale i vann, og kan operere i 20-30 timer.

2.2 Sensorer

Det finnes mange sensorer som er aktuelle for bruk på AUV-er, deriblant forskjellige varianter av avbildende sensorer. De tre vanligste som kan påmonteres HUGIN AUV er syntetisk aperture sonar (SAS), multistråle-ekkolodd og optisk kamera. SAS avbilder omgivelsene til sidene for farkosten, mens de to andre dekker området rett under. Det blir gitt en kort beskrivelse av disse tre sensorene i de følgende avsnitt.

2.2.1 HISAS 1030 syntetisk aperture sonar

HISAS 1030 er hovedsensoren til HUGIN. Det er en interferometrisk syntetisk aperture sonar, som avbilder robotens omgivelser med svært god oppløsning selv på store avstander (oppløsning er 3x3 cm ut til en avstand på 200 meter [18]). Ved bruk av tradisjonell side-scan sonar (SSS) må man typisk gjøre en avveining mellom lang rekkevidde og høy oppløsning i sonarbildet. SAS derimot, kan bruke en lav frekvens



Figur 2.2: *Synsfeltet til HISAS 1030 SAS. Illustrasjon fra [18].*

for å oppnå lang rekkevidde og deretter syntetisere en stor aperture ved å sette sammen data fra flere påfølgende ping. Dette gir en sensor som kan dekke store områder på kort tid, med svært god oppløsning [18].

Et eksempel på et SAS-bilde er vist i figur 2.1. Ved hjelp av slike bilder detekteres objekter på havbunnen (for eksempel ubåtvraket i figuren), og det er ofte mulig å klassifisere dem direkte ut fra sonarbildet. En forenklet illustrasjon over synsfeltet til HISAS 1030 er vist i figur 2.2. Sensoren kan også registrere informasjon om batymetri. Batymetri tilsvarer topografi, men beskriver forhold under vann. Det gir informasjon om bunnforholdene i havet eller en innsjø, for eksempel dybde eller grunnforhold (berg, sand etc.). I forbindelse med HUGIN AUV brukes batymetri om det å lage en terrengmodell av havbunnen. Detekterte objekter kan også inspiseres nærmere ved hjelp av de to andre sensorene, noe som krever at HUGIN navigerer bort og passerer direkte over dem.

2.2.2 Multistråle-ekkolodd

Den andre sensoren er et multistråle-ekkolodd, eller *Multi Beam Echo Sounder (MBE)*. Det er plassert på undersiden av roboten og produserer bilder som gir informasjon om avstanden til bunnen eller objekter på bunnen. Et eksempel er vist i figur 2.3(a). Multistråle-ekkolodd er en viktig og mye brukt sensor for å innhente informasjon om batymetri.

I masteroppgaven ville denne sensoren kunne blitt benyttet som hjelpesensor for det optiske kameraet (se neste avsnitt). Dersom man med det optiske kameraet har detektert strukturer som kan ligne et objekt, kan en praktisk anvendelse av multistråle-ekkolodd være å avgjøre om det antatte objektet stikker opp fra havbunnen eller ikke. Slik kan man med større sikkerhet skille faktiske gjenstander fra andre forekomster på havbunnen, for eksempel trålspor (slike spor kan sees i figur 2.1).

Kameraoppløsning	4008×2672 piksler (3×3 binning ofte brukt. Gir bildedimensjoner 1336×890 piksler)
Bilderate	$1 - 4$ Hz, avhengig av binning
Typisk hastighet, farkost	4 knop ($2^m/_s$)
Flyvehøyde, dårlig sikt	4 m (Kystfarvann og bynære fjorder om sommeren)
Flyvehøyde, god sikt	$8 - 10$ m (Åpent hav om vinteren eller tropiske dypvannsområder)

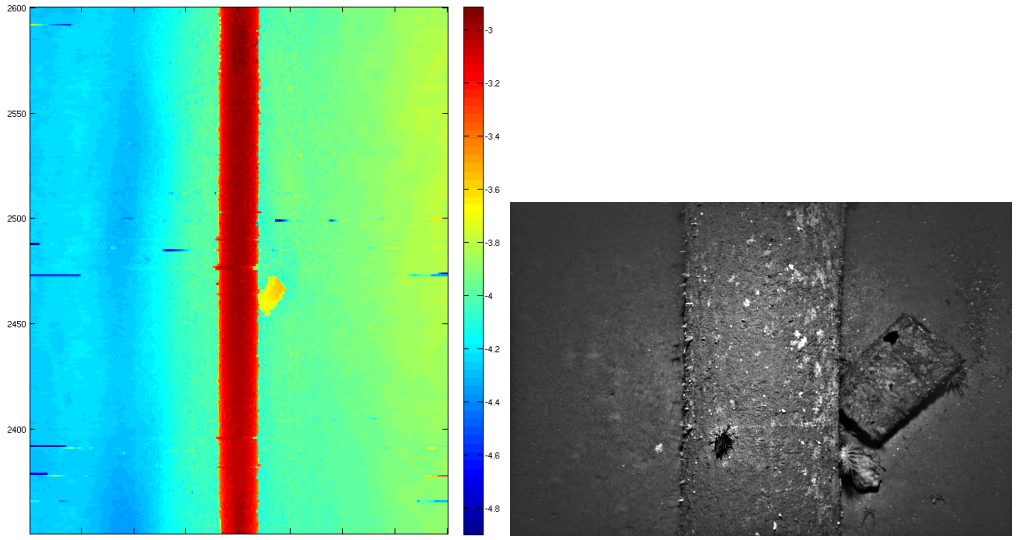
Tabell 2.1: *Relevante spesifikasjoner for HUGIN AUV ved bruk av «TileCam» optisk kamera.*

2.2.3 TileCam optisk kamera

HUGIN har også et spesialdesignet optisk stillbildekamera, *TileCam*, som fotograferer havbunnen under selve farkosten. Kameraet har blitt utviklet gjennom et samarbeid mellom Norsk Elektro Optikk (NEO), FFI og KM. Det produserer gråtonebilder med høy oppløsning, og har høy lyssensitivitet som gjør det egnet til å ta bilder i sparsom belysning. Kameraet er synkronisert med et strobelys laget av et panel med lysdioder (LEDs), som gjør det mulig å ta bilder der det ikke finnes naturlig lys. En høy bilderate gir også anledning til å danne sekvenser av overlappende bilder, eller til å simulere stereosyn [45]. Tabell 2.1 inneholder en oversikt over relevante spesifikasjoner for HUGIN AUV ved bruk av optisk kamera. Et eksempelbilde fra *TileCam* er vist i figur 2.3(b). På dette bildet har det blitt brukt homomorf filtrering for å øke kontrasten og for å kompensere for at strobelyset gir kraftigere belysning i sentrum av bildeflaten enn ute ved kantene. Denne filtreringsmetoden er beskrevet i avsnitt 6.2. Figur 2.4 viser plasseringen til kamera og LED-panel på roboten.

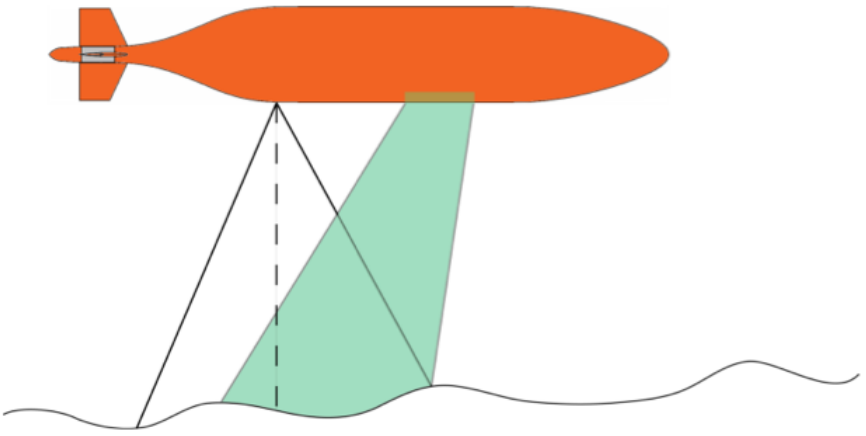
Å ta bilder under vann er ofte en utfordring, da det skal lite til før sikten blir dårlig. Vann vil absorbere og spre lyset i mye større grad enn luft, så sikten blir relativt kort. Grums i vannet reduserer bildekvaliteten ytterligere, så den lengste avstand det er mulig å se under vann vil typisk variere fra nær null til bare noen få titalls meter, avhengig av vannkvalitet og dybde. Dette er veldig korte avstander sammenlignet med for eksempel HISAS, som kan ha en rekkevidde på flere hundre meter. Multistråle-ekkoloddet kan også operere upåvirket av sikten i vannet og gir i tillegg nøyaktig informasjon om avstand til objektet. Grunnen til at man likevel ønsker å bruke optisk kamera er, når forholdene tillater det, fordi det avbilder objektet med høy oppløsning og i rask sekvens, og ikke minst fordi det viser objektet slik vi ser det med det blotte øye. Dette kommer tydelig frem ved sammenligning av de to bildene i figur 2.3, hvor kamerabildet gir et overlegent detaljnivå.

Bruk av optisk kamera på undervannsfarkoster kan være interessant i flere sammenhenger. Foruten det å avbilde enkeltobjekter, kan eksempel på bruk være å fotografere en rørledning som i figur 2.3(b), hvor man ønsker å inspisere tilstanden

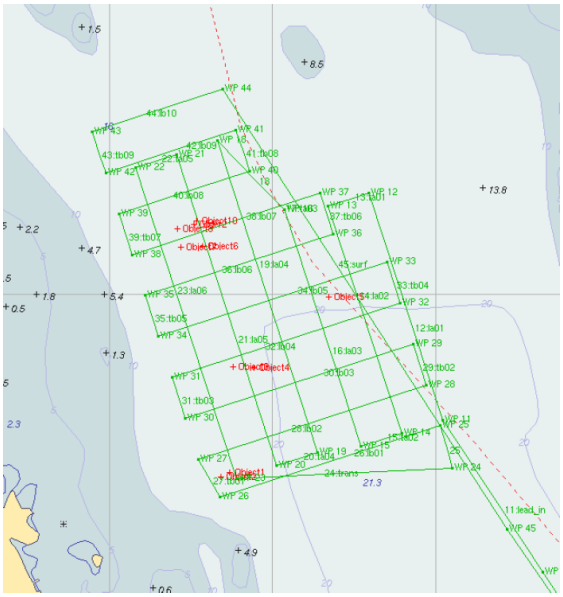


(a) *Bilde fra multistråle-ekkodlodd* (b) *Bilde fra optisk kamera (filtrert)*

Figur 2.3: *Rørledning og oljefat avbildet med både multistråle-ekkolodd (a) og optisk kamera (b). Høyde over bunnen kan leses av figur (a) til å være ca 4 m og avstanden til oljefatet til å være ca 3.5 m. Dette stemmer overens med diameteren til et standard 200 liters oljefat (57.2 cm [46]).*



Figur 2.4: *Montering og synsfelt til TileCam optisk kamera og LED-panel. Grønt felt symboliserer lysstråle fra LED-panelet. Illustrasjon hentet fra [30].*



Figur 2.5: *Mønster for kartlegging av et område med SAS. Her er bredden mellom linjene i ruteplanen (grønne linjer) 50 meter. [30]*

og avdekke skader på røret. Alternativt kan man i forbindelse med havforskning eller arkeologi ønske å ta overlappende bilder over et helt område, for eksempel for å danne et komplett bilde av et skipsvrak eller et korallrev.

2.3 Et typisk tokt for innsamling av bilder

Et eksempel på tokt hvor bruk av optisk kamera kan være nyttig, er mineryddingsoperasjoner. Målet er å undersøke et større område, avdekke eventuelle miner, bestemme hva slags miner som har blitt funnet og til slutt uskadeliggjøre dem. HUGIN AUV er med sine sensorer egnet for selve undersøkelsene, mens man i etterkant går inn med tradisjonelle metoder for å uskadeliggjøre eventuelle funn. Den følgende beskrivelse er basert på [16, 30, 33].

Toktet starter med en forhåndsprogrammert plan, hvor man avsøker et definert område med sonar. Farkosten kjører frem og tilbake i et såkalt gressklippermønster og samler kontinuerlig inn data fra omgivelsene. Fordi GPS ikke virker under vann er det vanskelig å opprettholde et godt estimat av egen posisjon. Ved å stige til overflaten kan man motta GPS-signaler og korrigere estimatet, men fra større dyp er dette upraktisk og blir likevel fort unøyaktig etter et nytt dykk. For HUGIN har det blitt utviklet avanserte teknikker til navigering og posisjonering under vann, og den er i stand til å gjennomføre et slikt tokt og følge en planlagt rute med god nøyaktighet. Et typisk mønster for avbildning av et område med sonar er vist i figur 2.5.

Det vanlige er at en AUV returnerer etter denne første kartleggingen slik at operatører kan analysere innsamlede data og lage en liste over mulige objekter.

Farkosten kan deretter sendes ut på et nytt tokt for å samle inn optiske bilder av de detekterte objektene. FFI har kommet langt i arbeidet med å la HUGIN gjøre denne operasjonen på egenhånd, slik at listen allerede er klar når området er ferdig avsøkt og at HUGIN selv kan lage en rute for å besøke hvert detekterte objekt. Til dette brukes en egen algoritme for ruteplanlegging som er tilpasset HUGINs manøveregenskaper (en modifisert «travelling salesman»-algoritme). Andre AUVer har benyttet seg av frontmontert sonar og videokamera, kombinert med videoanalyseteknikker som optisk flyt for å navigere mot detekterte objekter [32].

De innsamlede bildene brukes for å identifisere objektene, altså til å bestemme hva det er som har blitt tatt bilde av.

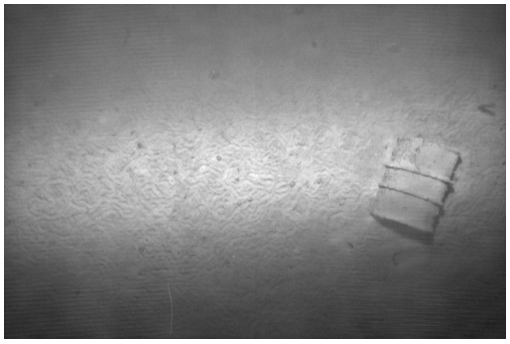
Kapittel 3

Introduksjon til prosessering av undervannsbilder

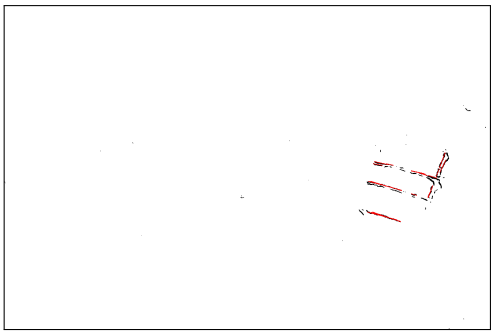
Etter at bildedata har blitt samlet inn som beskrevet i avsnitt 2.3, er det neste steget å analysere resultatene. Av de mange optiske bildene som har blitt tatt, kan det være svært få av dem som inneholder objekter av interesse. Et eksempel kan nevnes fra [23], hvor bilder blir tatt fortløpende gjennom et helt tokt. Av totalt 1317 innsamlede bilder er det bare 16 stk. som faktisk inneholder objekter. Å manuelt se gjennom alle bildene kan være en lang og tidkrevende prosess, noe som gjør dette til en jobb man gjerne ønsker å automatisere. Et mål er altså å la en datamaskin skille de interessante bildene fra de uinteressante (med så lav feilrate som mulig), noe artikkelen [23] også tar for seg. Der brukes en blokkbasert teknikk, som vil si at man deler opp bildet i mindre områder og analyserer delene hver for seg. Argumentasjonen for en blokkbasert teknikk er at bilder med og uten objekt skiller seg for lite fra hverandre ved bruk av globale teknikker, hvor man ser på hele bildet under ett. Problemet er at forgrunnens (objektenes) egenskaper «drukner» i den store mengden av bakgrunnsverdier.

Når man har funnet et bilde som faktisk inneholder noe av interesse, er neste oppgave å få laget en digital representasjon av objektet. Det kan være å segmentere ut hele objektet, eller å trekke ut spesielle egenskaper som er typiske for denne gjenstanden. Denne representasjonen kan videre sammenlignes med templater i en database for å avgjøre hva som har blitt funnet. Templatene kan være basert på bilder av virkelige objekter, eller på genererte 3D-modeller [31]. Å tilordne et funn til en modell av et kjent objekt kalles *klassifisering*.

En illustrasjon på prosessering av et bilde er vist i figur 3.1: Et oljefat kan kjennes igjen ved å se på omrisset eller ved å lete etter de tverrgående båndene på midten, noe som i stor grad har blitt detektert i figur 3.1(b). Disse har blitt funnet ved å lete etter rette linjer i bildet. At man får treff på de noe buede tverrbåndene skyldes robusthet og et slingringsmonn i detektoren. Et annet alternativ kunne være å se på tekstur, og dermed forsøke å skille oljefatets glatte overflate fra den antageligvis mindre glatte havbunnen. Dette vil være mindre aktuelt hvis objektet for eksempel har ligget lenge på et sted hvor mudder etter hvert dekker både havbunnen og objektets overflate.



(a) *Bilde fra optisk kamera (filtrert)*



(b) *Resultat av kantdeteksjon og terskling. Rette linjer er forsøkt detektert med houghtransform [15] (vist med røde streker i figuren).*

Figur 3.1: *De røde linjene i (b) er detekterte rette linjer, og kan brukes som typiske egenskaper til dette oljefatet. Ved klassifikasjon vil målet være å trekke ut egenskaper som vil matche ojefat-karakteristikker i en database.*

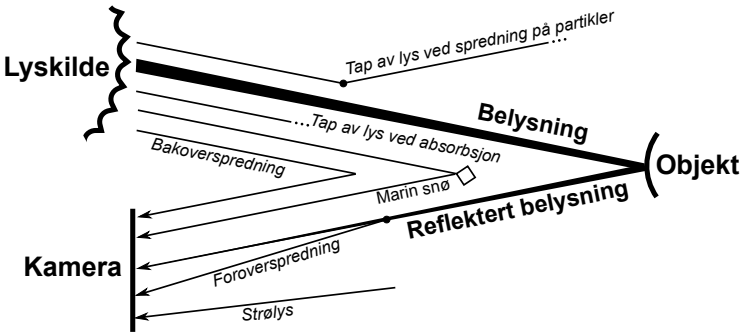
3.1 Problemer ved undervannsfotografering

I dette avsnittet omtales noen noen utfordringer knyttet til avbildning under vann. Innholdet er i hovedsak basert på [5, 31, 36].

Det å analysere undervannsbilder på jakt etter objekter på havbunnen byr på flere utfordringer. For det første kan gjenstandene kan være både tildekket og tilgrodd slik at de går i ett med omgivelsene og rett og slett er vanskelige å finne, og for det andre er det vanskelig å produsere gode bilder under vann, fordi fotografering baserer seg på lys. Under vann vil lys både absorberes og spres i mye større grad enn i luft, noe som vil begrense den avstanden det er mulig å se. En lysstråle som sendes gjennom vann vil dempes eksponensielt i forhold til avstanden den tilbakelegger. Dette gjør at scenen som avbildes blir tåkete og uklar. For å kompensere for absorbsjonen og gi lengre sikt, bruker HUGIN som tidligere nevnt en kunstig lyskilde. Figur 3.2 viser hvordan lyset går fra lyskilden, via objektet og tilbake til kameraet. Dette er hva vi ønsker å oppnå for å avbilde objektet, men som figuren også viser er det flere andre faktorer som spiller inn. Vannet i seg selv reflekterer en betydelig del av lyset tilbake til kameraet. Dette kalles bakoverspredning og effekten er at bildeflaten dekkes av et karakteristisk lysende slør som til en viss grad vil skjule scenen. Et annet fenomen er småvinkel foroverspredning, som beskriver tilfeldig spredning av lys på veg fra objektet tilbake til kameraet. Dette medfører at lyset treffer sensorbrikken litt til siden for der det egentlig skulle truffet, og dermed gjør bildet uskarpt.

Partikler i vannet er sammen med bakoverspredning den største årsaken til redusert bildekvalitet. Partiklene kan reflektere lyset bort og gjøre scenen mørk, og de kan bidra til foroverspredningen ved å forvrenge lys som reflekteres fra objektet.

I vann vil man også ofte observere større partikler av organisk materiale. Disse blir objekter i seg selv og avbildes som lyse prikker. Dette kalles «marin snø» fordi de lyse



Figur 3.2: *Problemer ved fotografering under vann.*

prikkene som oppstår ser ut som snøfnugg i bildet.

Strølys, det vil generelt si lys fra andre lyskilder som treffer kameraet direkte, vil også redusere kontrasten i bildet.

Dersom man tar fargebilder under vann, vil man ofte oppleve at bildene domineres av blått. Når vanlig hvitt lys går gjennom vann vil lys med ulike bølgelengder dempes forskjellig. Rødt lys dempes mest og blått lys minst. Dette er beskrevet i Beer-Lamberts lov [44], som generelt beskriver hvordan lys dempes i forhold til egenskapene til mediet det sendes gjennom. Et eksempel er vist i figur 3.3(a), hvor den blå dominansen kommer tydelig frem.

Problemer ved undervannsfotografering kan hittil oppsummeres med at avstanden det er mulig å se er relativt kort, kontrasten i bildene blir lav, bildene kan lett bli uskarpe, partikler og annet grums avbildes og skaper støy og i fargegjengivelsen vil blå ofte dominere. På HUGIN har man gjort noen valg for å kompensere for disse problemene. I det påmonterte strobelyset benyttes nettopp blågrønt lys for å minimere absorbsjonen i vannet og dermed gjøre det mulig å fotografere på så lange avstander som mulig. I tillegg er strobelyset og kameraet plassert i maksimal avstand fra hverandre på roboten, og hver lysdiode er ustyrt med en linse som styrer lysets retning. Ved å gjøre dette vil man redusere mengden lys som reflekteres inn i kameraet fra partikler i vannet, noe som i sin tur vil bedre kontrasten i bildene.

3.2 Forprosessering og objektgjenkjenning

I [25] omtales flere alternative teknikker for selve avbildningen i vann, typisk for å unngå de kjente problemene med kort sikt og dårlig kontrast. Det nevnes blant annet polarfiltrering for å unngå uønskede refleksjoner, og såkalte «range-gated methods». Ved gating-metoder benyttes en slags forsinket eksponering, hvor man sender ut et kort lysglimt og deretter åpner kameraet for eksponering først når man antar at lyset har rukket å bli reflekteret i objektet og kastet tilbake til kameraet. Slik kan man avbilde hva som finnes på flere ulike avstander og ved hjelp av dette kan man også lage 3D-modeller av scenen. Artikkelen tar også for seg andre områder innen

undervannsfotografering, som for eksempel lagring, kompresjon og plassering av kamera (på fartøy, stasjonært, festet på dyr, osv.). I masteroppgaven vil det arbeides med bilder fra allerede gjennomførte tokt, så selve innsamlingen av bildene er utenfor min kontroll.

På grunn av de vanskelige forholdene for avbildning, er det som regel behov for å gjøre en forprosessering av bildene slik at det senere blir lettere å analysere innholdet. I [36] blir det foretatt en gjennomgang av 15 forskjellige metoder for bildeforbedring. Teknikkene kan deles i to hovedgrupper. Den ene forsøker å rekonstruere «originalbildet» ved å modellere faktorene som har degradert det, for deretter å behandle bildene med det inverse av disse faktorene. Parametere kan for eksempel være dybde (for å kompensere for fargeavvik), sikt (graden av klart/grumsete vann) og så videre. Et problem med denne metoden er at slike faktorer kan være vanskelige å bestemme, og de kan være svært variable. Den andre teknikken er ren bildeforbedring, hvor man i stedet for rekonstruering forsøker å redigere bildet slik at man får det ønskede resultat. Den siste typen er som oftest enklere og raskere enn den første. Jeg velger å ikke gå i dybden på noen av teknikkene her, men et eksempel på forbedring av et fargebilde er vist i figur 3.3. Bildet illusterer godt hvordan ulike bølgelengder dempes forskjellig gjennom vannet. Som beskrevet i avsnitt 3.1 ser vi tydelig at den blågrønne fargen dominerer. Ved å kompansere for dette og fremheve de andre fargekomponentene i bildet gjennom prosessering, ser vi i figur 3.3(b) at motivet kommer mye tydeligere frem og at fargene blir mer naturlige.

(a) *Originalt bilde* (b) *Forbedret bilde*

Figur 3.3: *Kontrast og fargeforbedring av bilde ved bruk av teknikk fra Iqbal m.fl. Figurer hentet fra [20].*

Som tidligere nevnt har forprosesseringen til hensikt å tilrettelegge for videre bildeanalyse. Ofte brukt er algoritmer som reduserer støy uten å ødelegge viktige egenskaper i bildet som kanter og tekstur. Dette kan oppnås med forskjellige typer lavpassfiltrering. Både medianfiltrering og typetallfiltrering kan være egnet til å fjerne finkornet støy uten å ødelegge tekstur, kanter og kontraster [14]. En alternativ filtreringsmetode, anisotrop diffusjon [34], er brukt i [21]. Dette filteret glatter regioner og forsterker kanter ved å bruke en filtermaske som tilpasses informasjon lokalt i bildet.

I [17] gjennomføres bare en histogramutjevning for å dempe effekten av særlig lyse punkter som påvirker eksponeringen og gjør resten av scenen mørk.

Kanter er en egenskap som ofte brukes til objektgjenkjenning, da disse kan representere konturer eller karakteristiske linjer på objektet (som tidligere vist i figur 3.1). Det finnes flere filtre som kan trekke ut kanter i bilder, men Sobel og Laplace [15] er kanskje de vanligste. I [27] brukes fasekongruens for å detektere egenskaper (kanter) i bilder, noe som i artikkelen ser ut til å gi gode resultater. Houghtransform [15] er et aktuelt verktøy for å finne rette linjer eller sirkelbuer i bildene. Noen ganger vil slike regulære geometriske former på objektene maskeres, hvis for eksempel gjenstanden er delvis nedsunket i mudder på havbunnen. En mulighet for å gjenskape formene kan da være å benytte seg av regiongroing eller andre metoder for å slå sammen mindre grupper av sammenhengende piksler.

I [31] argumenteres det derimot for å unngå kantbaserte metoder til objektgjenkjenning, siden undervannsbilder lett kan domineres av støy og kontrastproblemer. Alternativet som blir presentert er en korrelasjonsbastert deteksjonsmetode, som rett og slett sammenligner to bilder og måler hvor like de er. Kamerabildet blir sammenlignet med datagenererte 3D-modeller av objektene man leter etter. Som nevnt i innledningen til dette kapittelet er det også mulig å bruke blokk-baserte teknikker for å detektere objekter. Det vil si at man ser på histogrammet i lokale deler av bildet hver for seg for å avdekke om det finnes et objekt der eller ikke. Histogrambaserte teknikker som ser på hele bildet under ett og ut fra dette forsøker å skille forgrunn og bakgrunn vil ofte fungere dårlig fordi objektenes farge gjerne går i ett med omgivelsene [14, 23].

På bakgrunn av en inspeksjon av havbunnsbilder som gjøres i neste kapittel, vil det gjøres et valg av metoder i kapittel 5.

Kapittel 4

Inspeksjon av havbunnsbilder

I dette kapittelet presenteres en samling bilder som har blitt brukt under arbeidet med oppgaven. Det er i hovedsak bilder fra treningssettet, det vil si bilder som har blitt brukt til eksperimentering underveis. Det er også inkludert et utvalg av bilder fra testsettet for å vise noe av variasjonen i datamaterialet. Testsettet brukes helt til slutt i oppgaven, for å måle ytelsen til detektoren som utvikles. Dette skal være uavhengig av treningssettet, slik at målet på ytelse blir realistisk i forhold til bruk av detektoren som en del av et system på en undervannsfarkost. Bildene er valgt ut med tanke på å vise ulike objekter, varierende vannkvalitet og variert havbunn. I første fase av eksperimenteringen bestod treningssettet av bare 6 bilder, men det ble etter hvert utvidet til 12 og senere 15 og 22 bilder for å få mer varierte treningsdata. 15 av bildene i treningssettet inneholder objekter. Testsettet består av totalt 157 bilder med tilsammen 92 menneskeskapte objekter fordelt på 83 av bildene.

Bildekarakteristikk

I dette avsnittet foretas en analyse av de presenterte bildene og objektene de inneholder. For oversiktens del er det valgt en løpende benevning av bildene med bokstaver $\{a, b, \ldots\}$, selv om bildene grupperes i ulike figurer som har hver sin figurtekst. Av den grunn fikk alle figurer i dette kapittelet nummeret 4.1.

Bildene (a) til (f) inneholder sylindriske objekter i ulik forfatning. Det er fem bilder av oljefat og ett med rester av en torpedo som ligger på mudderbunn. Det første og det fjerde, (a) og (d), viser i større grad en uskarphet som skyldes bakoverspredning av lys. Felles for alle er en relativt tydelig effekt av ujevn belysning som gir en mørk ring langs kantene av bildene.

Objektene i (g) og (h) ligger på en annen type havbunn. Først vises et betongelement på sand og steinbunn med noe vegetasjon omkring. Kontrasten mellom betongelementet og bakgrunnen er relativt dårlig, særlig den øverste og de venstre kantene. Badekaret som vises i (h) ligger på en eng av posidonia, en type sjøgress som blant annet vokser i middelhavsområdene. Kantene til dette badekaret har høy kontrast mot bakgrunnen.

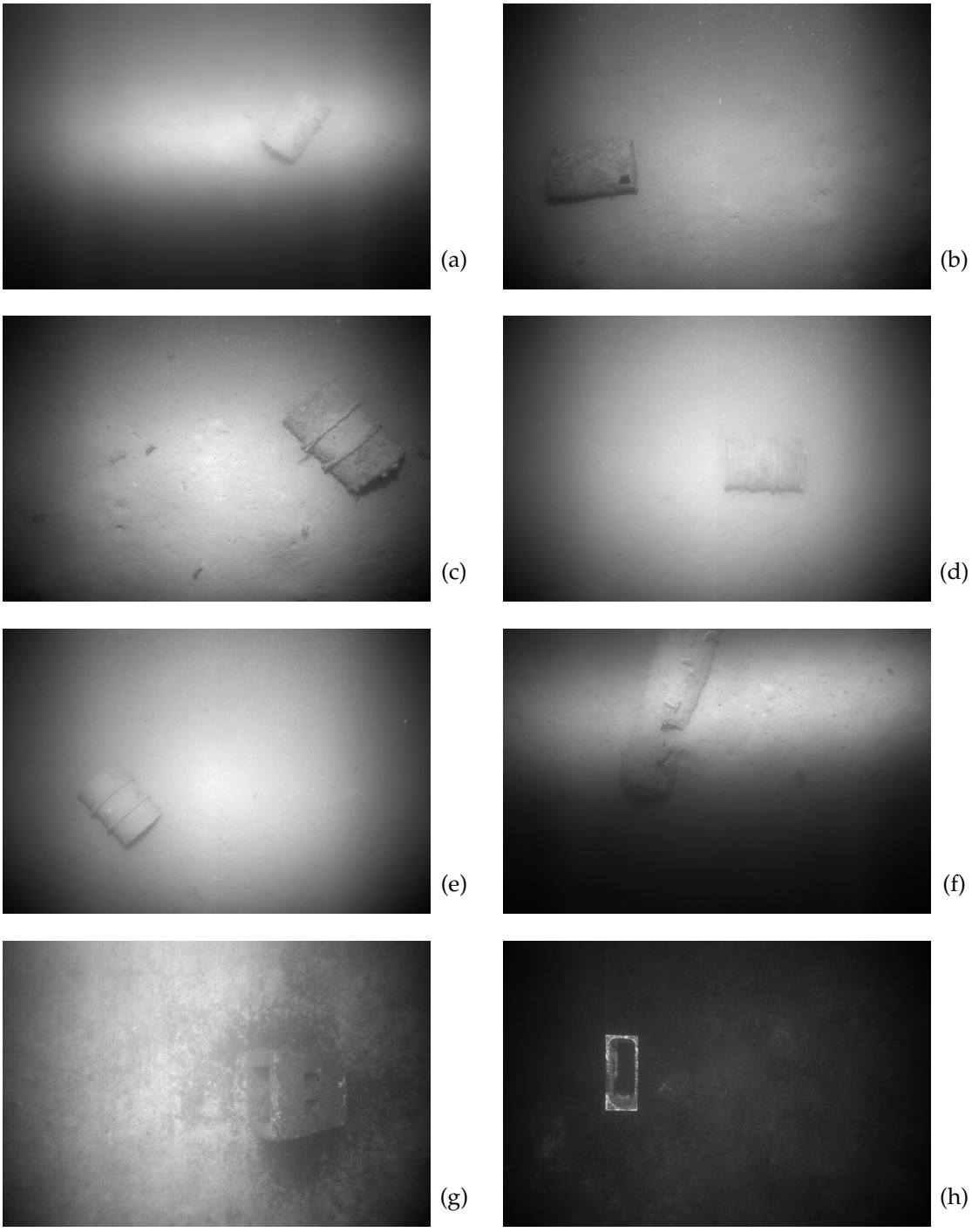
I neste gruppe vises fire bilder med ulike former for rør. I (i) er røret forankret til en rektangulær plate som stikker ut på hver side, omtrent midt i bildet. Her er det også endel vekster som dekker deler av platen. Kontrasten i bildet er forøvrig ganske redusert, på grunn av sterk refleksjon fra noe søppel eller et annet udefinert objekt som ligger i nedre høyre del av bildet. Dette vises som en hvit flekk, og gjør resten av bildet relativt mørkt. I (j) er det et tykkere rør som har bedre kontrast mot havbunnen. Dette kan skyldes at røret har en stor diameter og at det dannes skygger på sidene. Her vises ikke mye vegetasjon, men det er noen krepsehull som lager en viss struktur på havbunnen. Bildene i (k) og (l) har mye dårligere kontrast, og her er rørene også betydelig mindre. Rørene går mer i ett med havbunnen, men i noen områder oppstår det skygger som gjør det lettere å oppdage dem.

I (m) vises et bilde av et objekt som ser ut til å være menneskeskapt, men som ikke har noen typisk regelmessig form. Fasongen kan sies å være skål- eller panneformet, med kanter som stikker høyere enn objektet forøvrig. Bortsett fra dette kunne det også se ut som en stein. I dette utvalget er det også tatt med bildet (n) som viser en bag som er senket ned på havbunnen i forbindelse med en øvelse for undervannsfarkoster. Bildet er tatt med for å vise betydningen av flyvehøyde, det vil si avstanden fra farkosten til havbunnen. Da bildet ble tatt var sikten så god at HUGIN kunne gå relativt høyt og fortsatt avbilde havbunnen med god bildekvalitet. Den økte avstanden gjør at objektet ser lite ut, sammenlignet med for eksempel oljefatet som ble vist i (c). Bilde (o) viser et sirkelformet objekt som nesten går helt i ett med havbunnen. Denne er dekket av mye tynn vegetasjon som gir mye struktur. I tillegg er bildet tatt da objektet stod i en slik posisjon at lyskilden skaper få skygger, så objektet blir av flere årsaker vanskelig å oppdage.

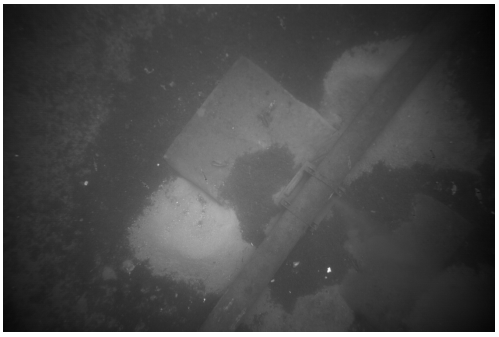
Bildene (p) til (r) er helt uten objekter. Det første av disse tre viser en sand og berggrunn der det har oppstått endel rette bruddflater i steinformasjonen. De to siste viser bare mudderbunn, (q) uten særlige kanter eller strukturer i det hele tatt, og (r) med en god del hull og spor etter kreps. I (r) er ikke belysningen bare ujevn med tanke på avstand fra sentrum, men hele nedre del av bildet er dårlig opplyst og får svært redusert kontrast.

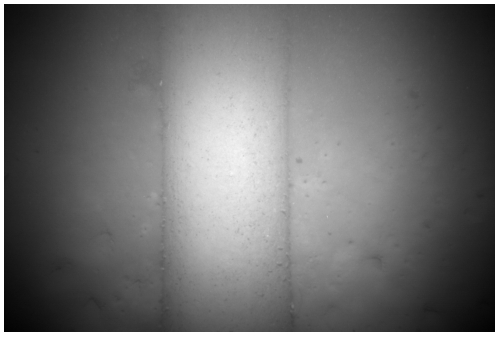
Oppsummering

- Bildene har generelt dårlig kontrast, og er sterkt påvirket av belysning.
- Type havbunn kan spenne fra sand til mudder, vegetasjon og fjell, som gir varierende grad av struktur på havbunnen.
- Dårlig kontrast og mye struktur på havbunnen gjør noen objekter vanskelige å se.
- Objektene vises i ulike størrelser og er i varierende forfatning. Noen er hele og uskadde, mens andre er delvis rustet og i stykker. Noen kan også være vanskelig å definere hva er. Det finnes både frittstående objekter og objekter som går fra kant til kant i bildet, og de finnes i ulike områder av bildeflaten.
- Objektene defineres og karakteriseres i hovedsak av sine konturer eller kanter, som forøvrig oftest trer frem på grunn av skygger.

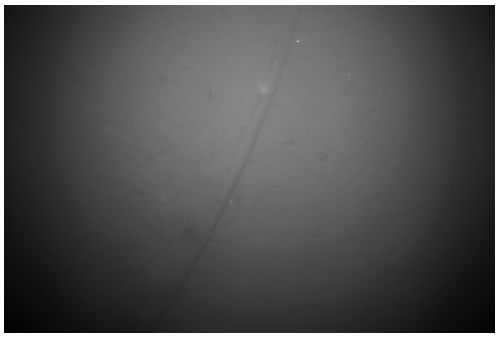


Figur 4.1: *Et utvalg menneskeskapte objekter i undervannsbilder.*


(i)

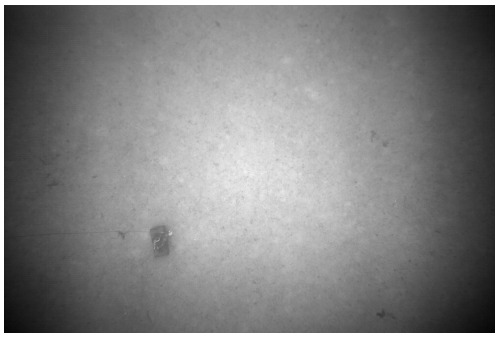

(j)

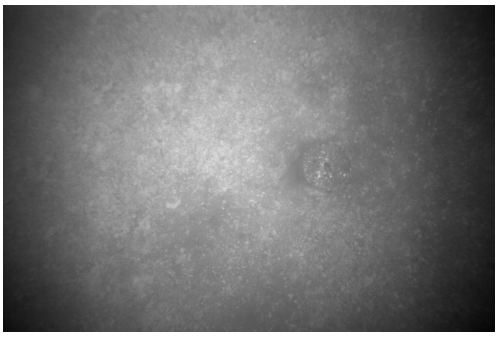

(k)


(l)

Figur 4.1: *Undervannsbilder av rør*

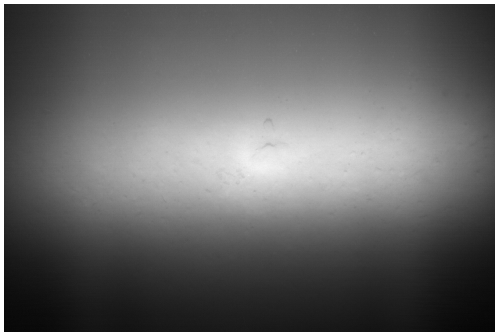

(m)


(n)


(o)

Figur 4.1: *Spesielle objekter*

(p)



(q)

(r)

Figur 4.1: *Bilder uten objekter*

Kapittel 5

Plan for oppgaven

For å detektere objekter i bildene må man ha noe bestemt å lete etter, noe som definerer objektet slik at det kan skilles fra andre elementer i bildet. Elementer i undervannsbilder som ikke er objekt vil i de fleste tilfeller være havbunn, og denne kan være svært varierende fra bilde til bilde. Det kan være steiner, sandrippler, vegetasjon eller mudder, og fisk og andre skapninger kan også bli med i bildene. Derfor bør egenskapene vi leter etter være noe som menneskeskapte objekter har til felles, slik at metoden kan brukes uavhengig av hva det har blitt tatt bilde av. Det er også ønskelig å velge disse egenskapene slik at vi kan detektere hele gjenstanden og dermed får med så mye informasjon som mulig når vi på et senere tidspunkt skal identifisere objektet.

Når vi mennesker ser på noe, vil blikket vårt naturlig trekkes mot kanter, konturer og kontraster. Dette har jeg valgt å også bruke som utgangspunkt for automatisk deteksjon, ved å lete i bildene etter mulige kanter som kan definere objekter. HUGIN AUV er som tidligere nevnt utstyrt med en hjelpebelysning for det optiske kameraet, og belysningen fører til at objekter som stikker opp fra havbunnen vil kaste skygger som kan sees i bildene. Dette gjør også at kanter som vender bort fra lyskilden vil tre frem, i den forstand at de kaster en skygge. En utfordring er at kanter og linjer også vil finnes som naturlige forekomster på havbunnen, men ved å stille krav til kantenes form og hvordan de ligger i forhold til hverandre, kan man anta at det er mulig å skille ut kanter som stammer fra menneskeskapte objekter. Når noe skal defineres som menneskeskapt kan en mulighet være å lete etter rene geometriske former som sirkler, sirkelbuer, rette vinkler, paralleller og rektangler.

Basert på karakteristikken av datasettet presentert i kapittel 4 har jeg hentet noen metoder fra litteraturstudien og valgt ut noen andre selv, som jeg tror kan være nyttige i arbeidet med oppgaven. Disse vil bli diskutert i kapitlene 6 og 7. Jeg vil prøve et par teknikker for å jevne ut bakgrunnsbelysningen og jeg ønsker å finne en metode for støyfiltrering. For deteksjon av objekter prøver jeg å se på både histogrambaserte og gradientbaserte metoder, men jeg vil også prøve fasekongruens som en alternativ kantdetektor. Til slutt ønsker jeg helt kort å se på tekstur som metode for objektdeteksjon. I eksperimentfasen har metodene blitt testet på flere ulike bilder, men i den følgende gjennomgangen er det bare brukt et fåtall for å illustrere effekten

av de ulike metodene. Metodene som testes ut er:

- Bottom-hat transform
- Homomorf filtrering
- Anisotrop diffusjon
- Global terskling og hysteresete terskling
- Adaptiv terskling med Bernsens metode
- Adaptiv terskling med Niblacks metode
- Sobel gradientoperator
- Cannys kantdetektor
- Houghtransform og radontransform
- Kantdeteksjon ved fasekongruens
- Teksturer ved førsteordens statistikk og GLCM

Kapittel 6

Forprosessering

Som beskrevet i avsnitt 3.1 er det flere utfordringer knyttet til undervannsfotografering. Lysstråler som går gjennom vann vil absorberes og spres – noe som fører til uskarpe bilder med dårlig kontrast – og bruk av kunstig lyskilde gir problemer i form av ujevn belysning. Store og små organiske partikler i vannet avbildes som lyse prikker og vil kunne «stjele» mye av dynamikken i gråtoneskalaen, slik at resten av scenen fortoner seg som mørk. At partiklene også blir objekter i bildet vil kunne medføre ekstra utfordringer når bildet skal analyseres på jakt etter menneskeskapte elementer.

I dette kapittelet beskrives utvalgte metoder for prosessering av bilder med tanke på å senere kunne gjøre et søk etter menneskeskapte objekter. Metodene er forsøkt beskrevet på en lettfattelig måte, og eksempler er vist på undervannsbilder fra HUGIN AUV. Følgende metoder blir vurdert:

- *Bottom-hat transform*, for å fremheve skygger og fjerne ujevn belysning,
- *Homomorf filtrering*, for å fjerne ujevn belysning,
- *Anisotrop diffusjon*, for kantbevarende glatting.

I tillegg omtales *geometrisk korreksjon*, som tas i bruk for å korrigere linseforvrengning.

6.1 Top-hat og bottom-hat, morfologisk bakgrunnsutjevning

En metode som kan brukes for å korrigere for ujevn bakgrunnsbelysning er såkalt *top-hat transform* [15]. Denne transformen kan også brukes både for å detektere - og for å fjerne strukturer av en viss størrelse, så det er rimelig å anta at den kan være nyttig i arbeidet med oppgaven. Den formelle definisjonen av top-hat transform er gitt av likning (6.1), hvor $\circ$ betegner morfologisk åpning av bildet f med strukturelementet b.

$$T_w(f) = f - (f \circ b) \tag{6.1}$$

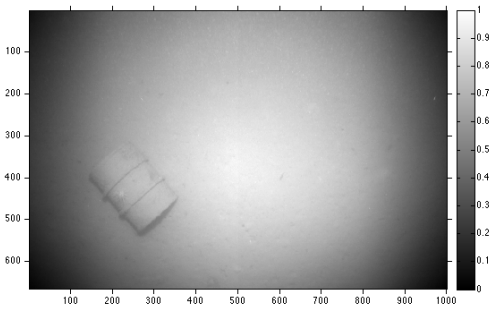
Ideen er at man ved morfologisk åpning fjerner elementer av interesse fra bildet, slik at man sitter igjen med en modell av bakgrunnen. Ved å subtrahere denne bakgrunnsmodellen fra originalbildet, vil differansen bare inneholde de objektene man

er ute etter. Prosesseringen gitt av likning (6.1) vil fremheve lyse områder i et bilde og kalles gjerne *hvit top-hat transform*. Tilsvarende kan man utføre en *svart top-hat transform*, eller en *bottom-hat transform* for å detektere mørke områder. Dette er definert som differansen mellom det morfologisk lukkede bildet og originalbildet:

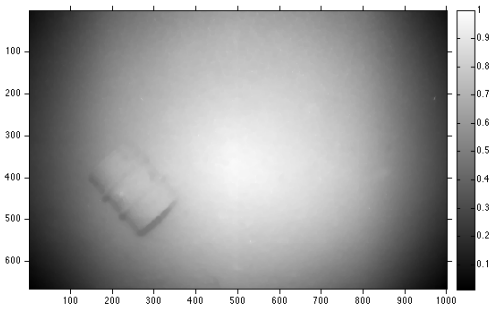
$$T_b(f) = (f \bullet b) - f \tag{6.2}$$

Størrelse og form på strukturelementet er avgjørende for hvilke strukturer som detekteres. Dette gjør at man for eksempel kan fjerne små objekter fra bildet uten at større strukturer endres, men det betyr også at deteksjonen blir mislykket dersom det velges et strukturelement som ikke passer til problemet som skal løses. Prosessen for bottom-hat er illustrert steg for steg i figur 6.1. Figurene 6.1(b) til 6.1(d) viser resultat av lukking ved bruk av forskjellig størrelse på strukturelementet. For en vellykket transform er det ønskelig å fjerne alle skyggeelementer før subtraksjonen, men selv med det relativt store strukturelementet som er brukt i figur 6.1(d), et sirkelformet strukturelement med radius 21 piksler, er det fortsatt mulig å skjelne skyggen av oljefatet. Ideelt sett skulle denne forsvinne fullstendig, så vi vil ikke få optimal effekt av bottom-hat transform på dette eksempelet. Skyggedeteksjonen blir likevel relativt god, fordi det finnes en skarp kant mellom oljefatet og dets skygge i originalbildet. Når denne smøres tilstrekkelig ut slik det er gjort i figur 6.1(d), vil subtraksjonen gjøre at kanten bevares i den endelige bottom-hat transformen, slik vi ser i figur 6.1(e).

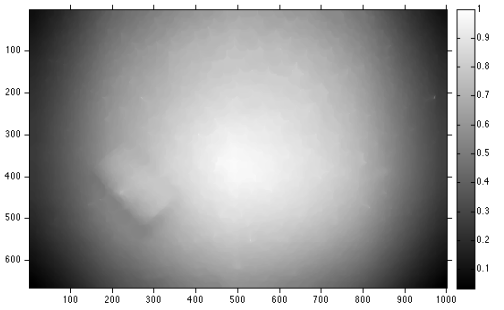
I originalbildet kan vi se organiske partikler i vannet avbildet som små, hvite prikker. Fordi morfologisk lukking vil tilordne et piksel den lyseste gråtoneverdien under strukturelementet, og fordi de organiske partiklene er relativt godt spredd, vil vi få et slags skjoldete resultat av åpningen. Dette viser seg i form av ring-effekter i både figur 6.1(d) og 6.1(e). Med et mindre strukturelement blir det færre ring-effekter i det homogene området, og dermed også færre falske skygger og mindre støy ved segmentering fra terskling. Samtidig blir det mindre tydelige skygger rundt objektet, som gjør det nødvendig med en høyere terskel for å detektere skygge. Da introduseres igjen mer støy i bildet. Forsøk på å filtrere bort disse partiklene er beskrevet i avsnitt 7.1.2.



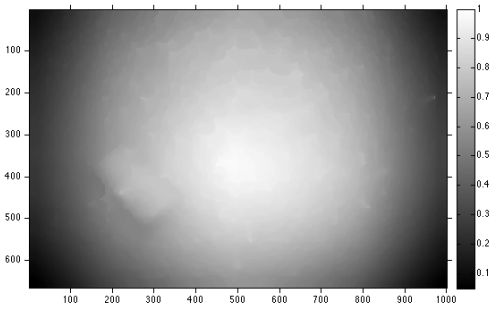
(a) *Originalbilde.*



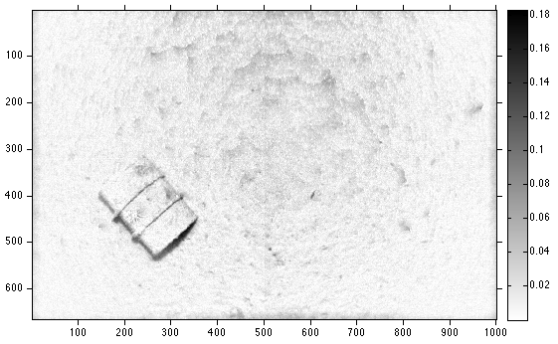
(b) *Morfologisk lukking med strukturelement* `strel('disk',5)`



(c) *Morfologisk lukking med strukturelement* `strel('disk',15)`



(d) *Morfologisk lukking med strukturelement* `strel('disk',21)`



(e) *Bottom-hat transform med strukturelement* `strel('disk',21)`*, invertert fargeskala for bedre visning på papir.*

Figur 6.1: *Morfologisk åpning med forskjellige strukturelementer og til slutt resultatet av bottom-hat transform. Med riktig størrelse på strukturelementet vil mørke partier fremheves, mens forskjeller i bakgrunnsbelysning jevnes ut.*

6.2 Homomorf filtrering

Ved undervannsfotografering er det som regel alltid behov for en form for ekstra lys, siden dagslyset raskt absorberes i vannet. Dette benyttes også på HUGIN AUV, som har produsert bildene som brukes i oppgaven. Med en ekstra lyskilde kommer det også noen nye utfordringer, blant annet kan vi få problemer med ujevn belysning i bildene. Dette gjør at de blir mørke i kantene og gradvis lysere inn mot sentrum. Et eksempel er vist i figur 6.2(a). Informasjonen i dette avsnittet er basert på [13, 15, 38].

I teorien om hvordan bilder dannes, brukes i blant en modell som kalles *belysning og reflektansmodellen*. Den sier at intensiteten i ethvert piksel, som varierer med mengden lys som reflekteres fra et punkt i scenen, er et produkt av belysningen som opplyser motivet og reflektansen[1] fra objektene som avbildes. Hvis $f(x,y)$ er pikselintensitet og $i(x,y)$ og $r(x,y)$ uttrykker henholdsvis belysning og reflektans, er uttrykket gitt som i likning (6.3):

$$f(x,y) = i(x,y)r(x,y) \tag{6.3}$$

Siden reflektansen hovedsaklig er en egenskap knyttet til selve objektene i scenen og belysningsleddet stammer fra lysforholdene da bildet ble tatt, kan vi kompensere for ujevn belysning ved å bare justere belysningsleddet, altså modifisere $i(x,y)$.

Ved å anta at belysningen endrer styrke på en gradvis og relativt langsom måte over bildeflaten, mens reflektansen karakteriseres av bråere overganger og kraftige endringer rundt objekter i scenen, kan det være logisk å foreta en Fouriertransform [15] for å skille ulike frekvenser i bildet, og dermed skille belysning og reflektans. En høypassfiltrering av bildet i frekvensdomenet vil da være hensiktsmessig for å fjerne de lavfrekvente bidragene fra belysningsleddet. Men, siden en multiplikasjon i bildedomenet ikke er en multiplikasjon i frekvensdomenet gjøres det først en log-transform slik at vi får additive komponenter og videre kan behandle belysning og reflektans hver for seg. Vi har at

$$\begin{aligned} f(x,y) &= i(x,y)\ r(x,y) \\ \ln f(x,y) &= \ln(i(x,y)\ r(x,y)) \\ \ln f(x,y) &= \ln i(x,y) + \ln r(x,y) \\ \mathcal{F}[\ln f(x,y)] &= \mathcal{F}[\ln i(x,y)] + \mathcal{F}[\ln r(x,y)], \end{aligned}$$

hvor $\mathcal{F}$ indikerer Fouriertransformen til bildet. Notasjonen forenkles ved å si at $\mathcal{F}[\ln g(x,y)] = F_g(x,y)$ for en vilkårlig funksjon $g(x,y)$. Vi får

$$F_f(x,y) = F_i(x,y) + F_r(x,y) \tag{6.4}$$

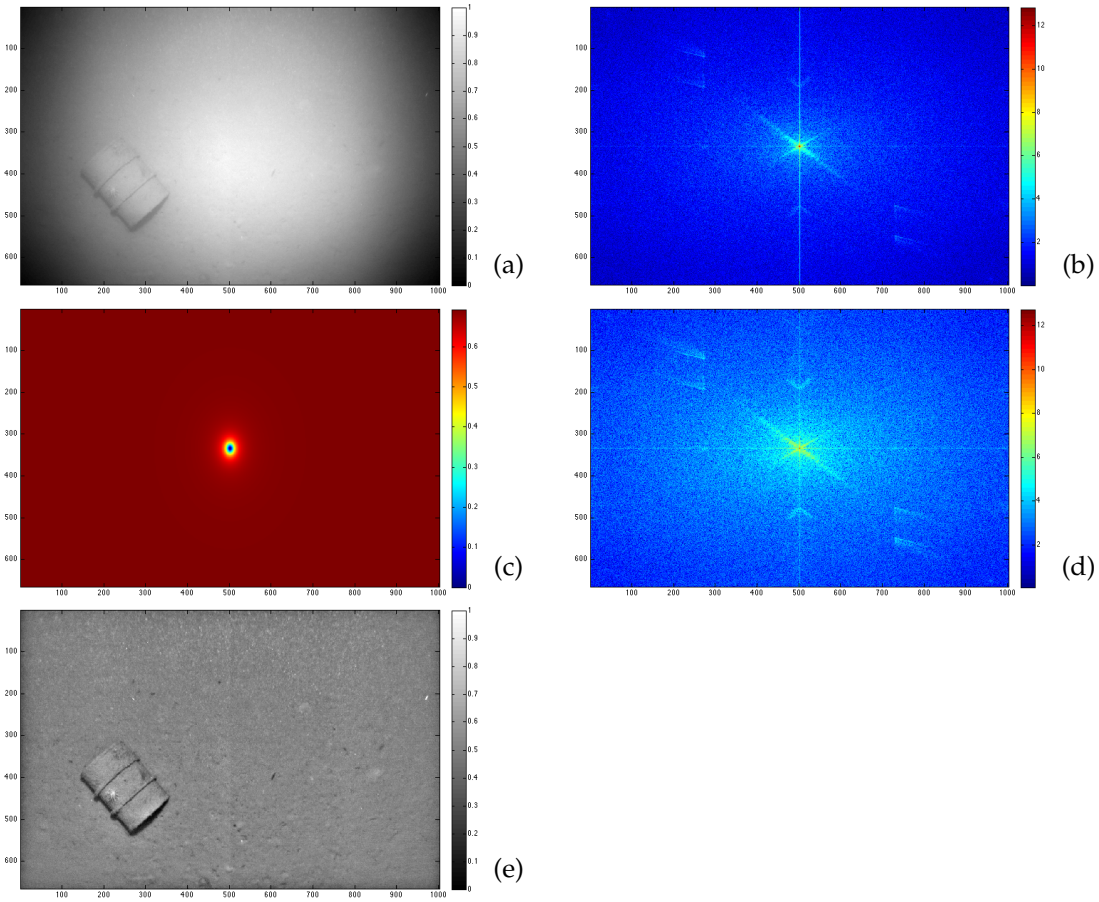
Ved bruk av belysnings- og reflektansmodellen er det nå kjent at $F_i(x,y)$ for det meste vil inneholde lave frekvenser, mens $F_r(x,y)$ vil inneholde høyere frekvenser. Hvis vi nå høypassfiltrerer $F_f(x,y)$ er det dermed rimelig å anta at vi fjerner mesteparten

[1] reflektans: reflektert stråling

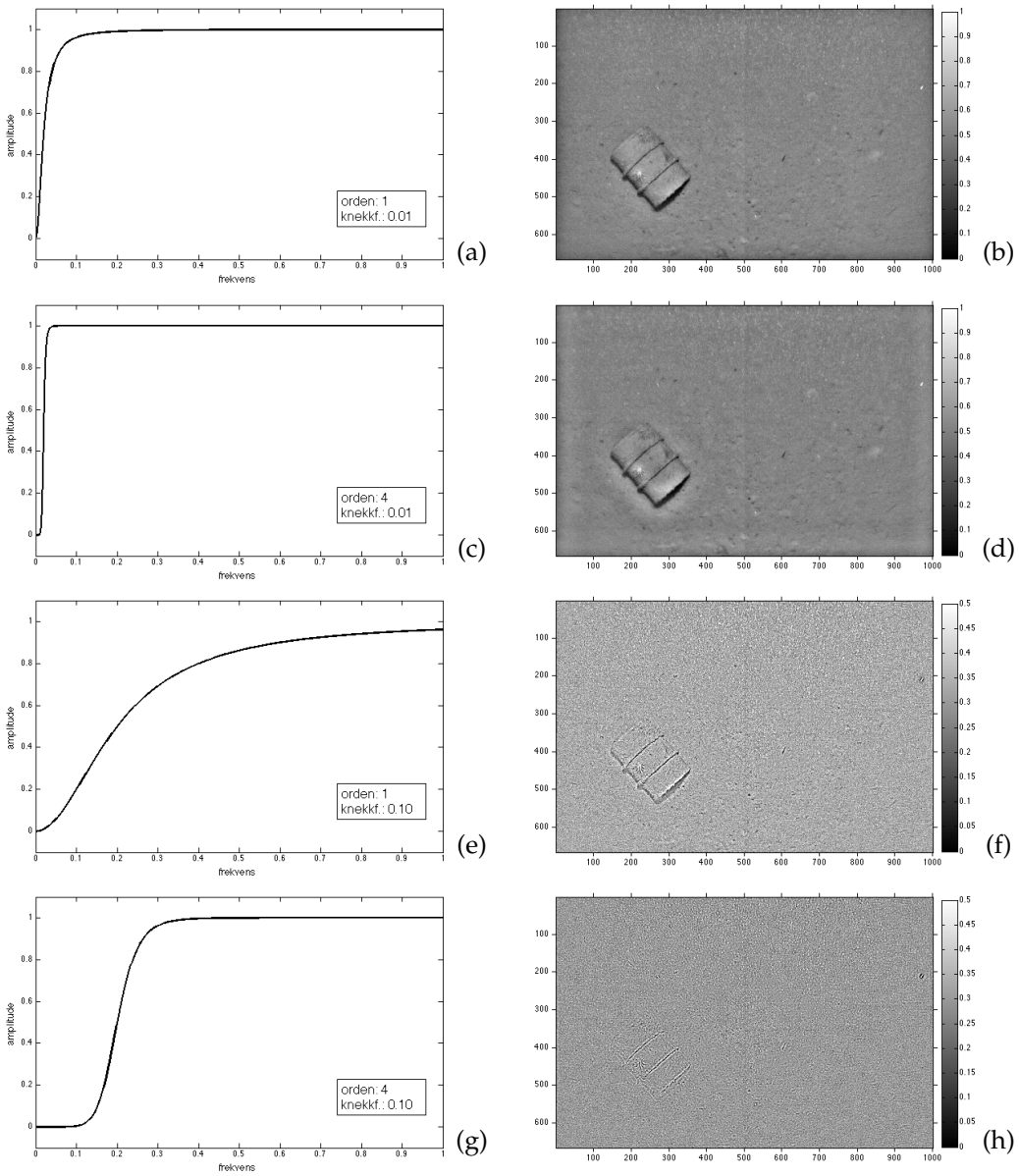
av bidraget fra belysningen $F_i(f,x)$ mens bidraget fra reflektansen $F_r(f,x)$ forsterkes. Gjennom dette oppnås en utjevning av belysningen samtidig som lokal kontrast i bildet øker.

Effekten av homomorf filtering er vist i figur 6.2. Figur 6.2(a) viser det ufiltrerte bildet, hvor ulempene ved kunstig belysning kommer tydelig frem. Bildet er lysest i midten og blir gradvis mørkere mot kantene og hjørnene, hvor det er tilnærmet ingen visuell informasjon. Fourierspekteret [15] til bildet er vist i figur 6.2(b). Denne multipliseres med Fouriertransformen til filteret, vist i figur 6.2(c). Dette har lavest verdi (kraftigst dempning) for 0-frekvensen midt i plottet, og stiger raskt til verdier som gir liten dempning for de høyere frekvenser. Resultatet av multiplikasjonen er vist i figur 6.2(d), og konvertert tilbake til bildedomenet blir resultatet som vist i figur 6.2(e) der vi kan se stor forbedring. Belysningen har blitt korrigert og kontrasten er klart forbedret. Dette bildet vil være mye bedre egnet for videre analyse.

`MATLAB`-implementasjonen av filteret som er brukt i oppgaven er vist i programkode A.1 i vedlegg A. Filteret som brukes er et Butterworthfilter med justerbar orden og knekkfrekvens. Knekkfrekvensen bestemmer hvilke frekvenser som dempes av filteret, mens filterets orden bestemmer hvor bratt filteret knekker. Effekten av forskjellige parameterinnstillinger for filteret er vist i figur 6.3. På øverste rad vises frekvensrespons og resultat av filtrering slik det er brukt i oppgaven. Filteret operer med normaliserte frekvenser fra 0 til 1, og å sette denne til 0.01, det vil si å fjerne de 1% laveste frekvensene i bildet, har fungert bra. Det fungerte også best å bruke første orden, eller orden 1. Dette er vurdert ut fra empiri. Å bruke en høyere orden på Butterworthfilteret vil gi antydninger til ringningsffekter, slik det er mulig å se omkring oljefatet i (d). Dette fenomenet er en følge av at filtreringen blir gjort i frekvensdomenet. En høyere knekkfrekvens vil filtrere bort en større del av de lavfrekente komponentene i bildet, slik som vist i (f) og (h). Her ser man at det fjernes verdifull kontrast rundt kantene av objektet, og at de små lyse partiklene som finnes i øvre halvdel av bildet bevares.



Figur 6.2: *Eksempel på homomorf filtrering. Belysning jevnes ut og lokal kontrast økes ved å dempe de laveste frekvensene i Fourierspekteret. Figurene (a) til (e) viser gangen i filtreringen, fra bildedomenet til frekvensdomenet og tilbake igjen. Originalbildet i (a) har typiske kjennetegn på ujevn belysning. I (b) til (d) vises henholdsvis Fourierspekteret til bildet, filteret og det filtrerte bildet. Filtreringen utføres ved å gjøre en punktvis multiplikasjon av det Fouriertransformerte bildet og filteret. Til slutt i figur (e) vises det filtrerte bildet konvertert tilbake til bildedomenet. Figurene som viser Fourierspektra er konvertert til logaritmisk skala for bedre visning av magnitude.*



Figur 6.3: *I venstre kolonne vises frekvensresponsen til et Butterworth høypassfilter ved forskjellige instillinger for orden og knekkfrekvens. Høyre kolonne viser resultatet etter filtrering med det sidestilte filteret. I oppgaven har det blitt brukt innstillinger som vist i (a) og (b). Høyere knekkfrekvenser fjerner mer og mer informasjon fra bildet, og i (f) og (h) (frekvens 0.10) er nesten all kantinformasjon fjernet. Gråtoneskalaen går fra 0 til 1, men er komprimert her for noe bedre visning på papir.*

6.3 Anisotrop diffusjon

Anisotrop diffusjon [34] er en støyfiltreringsteknikk som søker å redusere støy i et bilde uten å fjerne viktige detaljer som kanter, linjer, eller andre elementer som er viktige for tolkningen av bildet. Det betyr at glatting i større grad utføres innad i regioner enn på tvers av kanter mellom regioner. Tenk for eksempel på et bilde av et tre mot himmelen. En normal glatting vil smøre det grønne bladverket og den blå himmelen ut i hverandre. Anisotrop diffusjon vil i motsetning til dette jevne ut bladverket til en grønn region og himmelen til en blå region uten at grensen mellom dem blir utydeliggjort.

Grunnlaget for prosessen er en iterativ glatting, hvor man genererer et sett av bilder der hvert nye bilde er en glattet versjon av det forrige. Dette er sammenlignbart med isotrope diffusjonsprosesser i fysikken, som beskriver hvordan temperatur, trykk eller konsentrasjon av stoffer vil utlignes over tid. I denne filtreringen tar man ikke vare på alle bildene i settet, men man gjentar prosessen til en tilfredsstillende grad av glatting er oppnådd.

Anisotropi beskriver noe som har ulike egenskaper avhengig av retning. For å eksemplifisere begrepet kan man tenke på treverk, som er lettere å dele langsetter veden enn på tvers. I tilfellet med anisotrop diffusjon betyr det at graden av diffusjon/glatting avhengiger av egenskaper i bildet, og ikke nødvendigvis er lik i alle retninger.

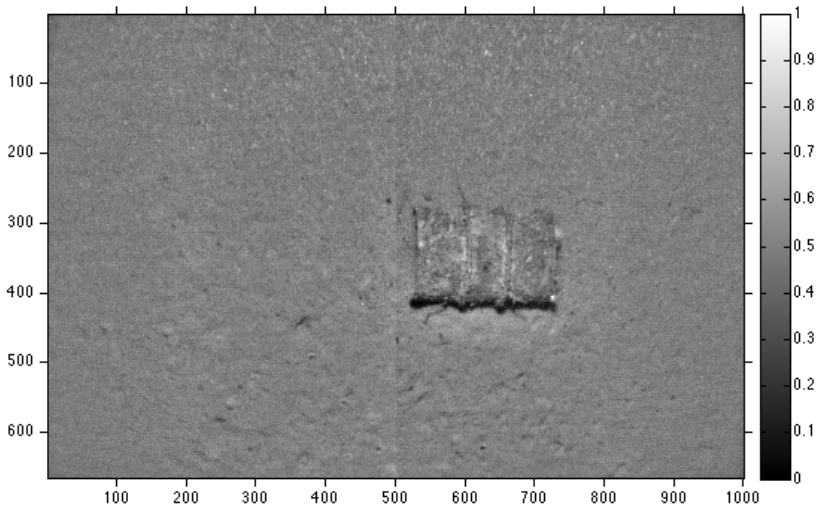
Dette implementeres som et konvolusjonsfilter, hvor filterkjernen i det generelle tilfellet utfører glatting ved hjelp av et gaussfilter. I hvert vindu gjøres det en beregning av lokal gradient, og i områder av bildet som inneholder kanter eller linjer vil det i stedet for en gauss brukes en filterkjerne som er tilpasset retningen til denne strukturen. Dette gjør at det utføres en glatting langsetter, men ikke på tvers av strukturen [21, 34, 43].

I implementasjonen jeg har brukt [26] er et piksels verdi i neste iterasjon gitt som en vektet funksjon av dets fire nabopiksler. Relativt like nabopiksler vil nærme seg hverandre i verdi, mens nabopiksler med stor differanse vil i større grad beholde sine opprinnelige verdier. Dette blir gjort gjennom oppdateringsfunksjonen i likning (6.5). λ kontrollerer hastighet til diffusjonen, og settes gjerne til maksimalverdien 0.25. Δ_x hvor $x = \{N, S, E, W\}$ indikerer differanse i pikselintensitet i henholdsvis retning nord, sør, øst og vest, mens c_x er diffusjonskoeffisienten, eller vektingen av differansen. Det har blitt foreslått to funksjoner [34] for c_x, gjengitt i likning (6.6). $\kappa \in [0, 1]$ kontrollerer sensitiviteten i forhold til kanter. Hvis κ er lav vil små intensitetsdifferanser tolkes som kanter og hindre diffusjon. En høy κ tillater glatting på tvers av høyere differanser og dermed glatting av tydeligere kanter.

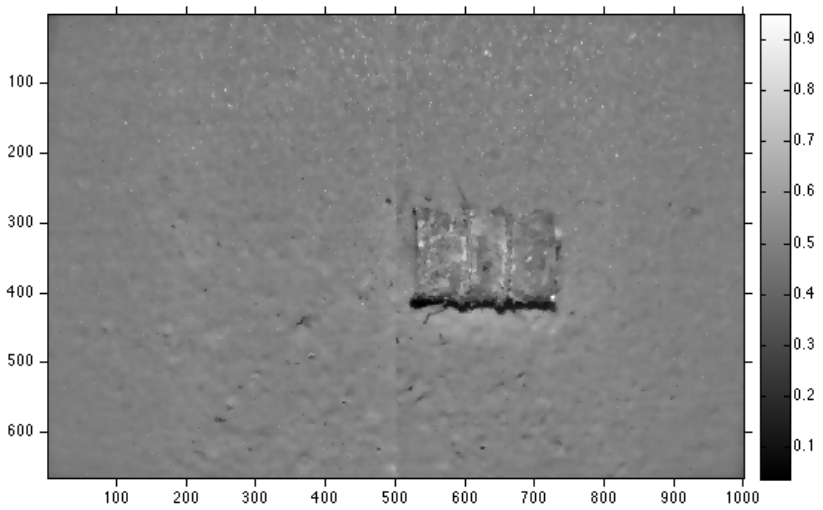
$$I_{i+1} = I_i + \lambda \cdot (c_N \cdot \Delta_N + c_S \cdot \Delta_S + c_E \cdot \Delta_E + c_W \cdot \Delta_W) \tag{6.5}$$

$$c_x = exp(-(\Delta_x/\kappa)^2) \tag{6.6a}$$

$$c_x = 1/(1 + (\Delta_x/\kappa)^2) \tag{6.6b}$$

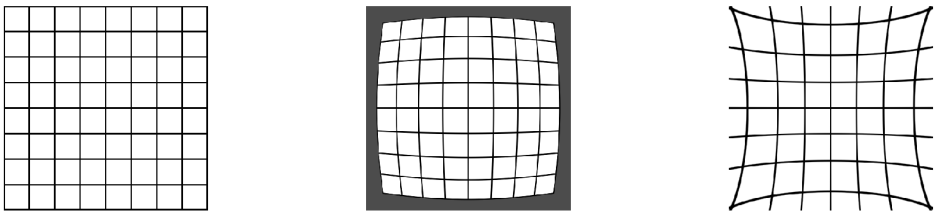


(a) *Bilde filtrert med homomorf filtrering*



(b) *Resultat etter 20 iterasjoner med anisotrop diffusjon. Diffusjonskoeffisient gitt av likning* (6.6a), $\kappa = 0.04$, $\lambda = 0.25$.

Figur 6.4: *Filtrering med anisotrop diffusjon. Den homogene havbunnen og flatene på oljefatet glattes betydelig, mens kantene på objektet bevares. Legg også merke til at «marin snø» med høy kontrast til bakgrunnen også har blitt bevart i øverste del av bildet.*



(a) *Rutenett uten forvrengning* (b) *Tønneforvrengning* (c) *Nåleputeforvrengning*

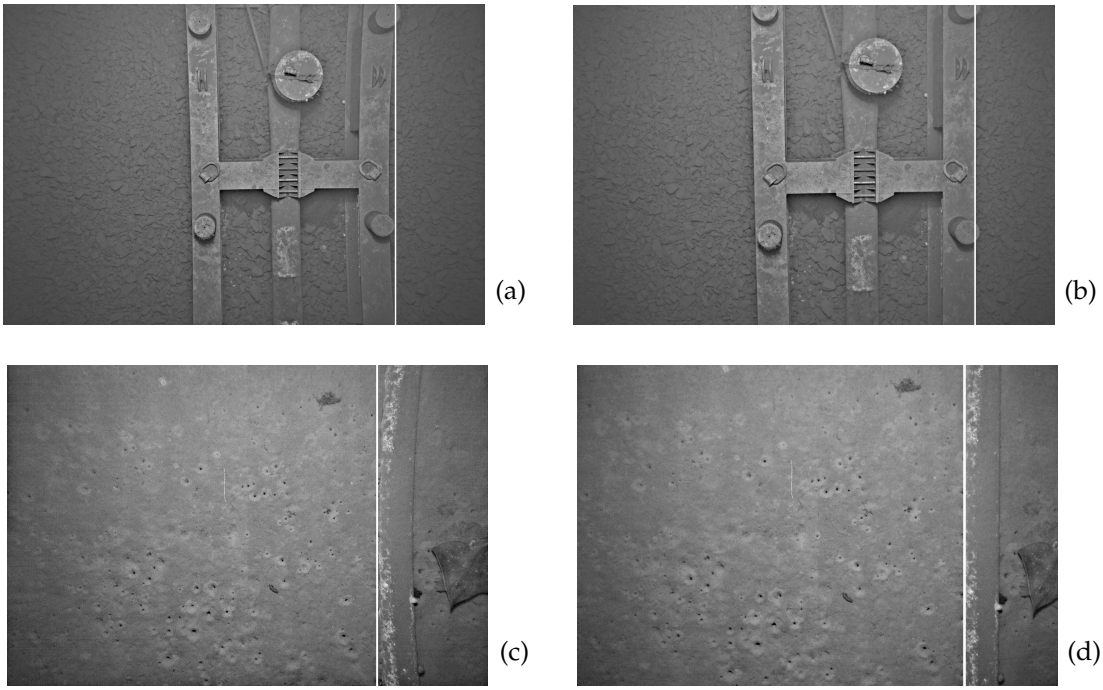
Figur 6.5: *Hovedtyper av radiell forvrengning.*

6.4 Geometrisk korreksjon

Ved all fotografering som innebærer bruk av linser er det sannsynlig at såkalt linseforvrengning inntreffer i større eller mindre grad. Forvrengningen fører til at linjer som er rette i virkeligheten kan se bøyd ut i bildet, og effekten er vanligvis sterkere jo nærmere kanten av bildet linjen ligger. Eksempler som vil være kjent for mange er fotografering med vidvinkelobjektiver eller «fish eye»-linser, som ofte gir svært kraftig forvrengning. Når man i bilder leter etter objekter som defineres av rette kanter og vinkelrette hjørner, er det åpenbart at en slik forvrengning vil være uheldig.

Problemet oppstår fordi linsen kan ha kraftigere (eller svakere) grad av forstørring i sentrum enn ute ved kantene. Dersom forstørringen er kraftigst i sentrum av linsen oppstår en «tønneforvrengning» *(barrel distortion)* og dersom forstørringen er kraftigst i kantene av linsen oppstår en «nåleputeforvrengning» (*pincushion distortion*). Disse typene av forvrengning kalles med en fellesbetegnelse for radiell forvrengning og er vist med eksempler i figur 6.5 [41]. For å veie opp for dette problemet kan man foreta en såkalt geometrisk kamerakalibrering som gjør det mulig å utføre geometrisk korreksjon av bilder. Implementasjonen av geometrisk korreksjon for kameraet på HUGIN AUV har ikke vært en del av arbeidet med selve masteroppgaven og er derfor ikke omtalt i detalj her, men en beskrivelse kan leses i [37]. For en fullstendig beskrivelse av teori og prosedyre for kamerakalibrering henvises det til [19, 40].

To undervannsbilder før og etter korreksjon er vist i figur 6.6. Ved hjelp av linjene som er lagt over figurene kan man se en tydelig nåleputeforvrengning i bildene til venstre, og at forvrengningen er rettet opp i bildene til høyre. Det kan også sees at de opprettede bildene er såvidt beskåret. Den radielle distorsjonen øker proporsjonalt med avstanden fra sentrum av bildet, og ved korreksjon har derfor pikslene i hjørnene blitt flyttet lengst tilbake inn mot midten. Dette gir det korrigerte bildet en ellipselignende form, og det foretas en beskjæring til det største rektangelet som passer innenfor ellipsen.



Figur 6.6: *Geometrisk korreksjon av undervannsbilder. Venstre kolonne viser bilder før korreksjon, og høyre kolonne viser resultatet etter korreksjon. Hvite vertikale linjer er lagt over figurene for å tydeliggjøre virkningen. Bildene er behandlet med homomorf filtrering.*

Kapittel 7

Segmentering av objekter på sjøbunnen

I de følgende avsnitt blir det sett på ulike tilnærminger for deteksjon av objekter på havbunnen. Hensikten er å teste flere metoder og evaluere deres styrker og svakheter uten å gå i dybden for hver enkelt metode. Ulike parametervalg er hardkodet for testingens skyld, der ikke annet er spesifisert. I eksperimentfasen ble metodene testet på flere ulike bilder, men i dette kapittelet er det bare brukt et fåtall for å illustrere effekten av de ulike metodene.

Avsnitt 7.1 handler om skyggedeteksjon og ulike former for terskling, mens avsnitt 7.2 omtaler gradientbaserte metoder. I avsnitt 7.3 diskuteres bruk av houghtransform og radontransform for linjedeteksjon, og i avsnitt 7.4 blir fasekongruens gjennomgått. I avsnitt 7.5 blir det kort testet bruk av tekstur som egenskap for objektdeteksjon. Til slutt vurderes resultatene i avsnitt 7.6 og det gjøres et valg for hvilke metoder som brukes videre i oppgaven.

Temaer som omtales i kapittelet er:

Skyggedeteksjon

- Global terskling
- Global terskling med bakgrunnsutjevning (bottom-hat og homomorf filtrering)
- Global terskling med bakgrunnsutjevning og støyfiltrering
- Hysteresetерskling
- Otsus metode (ikke brukt her, bare beskrevet)
- Adaptiv terskling med Bernsens metode
- Adaptiv terskling med Niblacks metode

Kantdeteksjon

- Sobel gradientoperator
- Kantbevarende glatting med anisotrop diffusjon

- Cannys kantdetektor
- Utfordringer knyttet til valg av terskelverdi for gradientbilder
- Houghtransform og radontransform av gradientbildet
- Kantdeteksjon ved fasekongruens

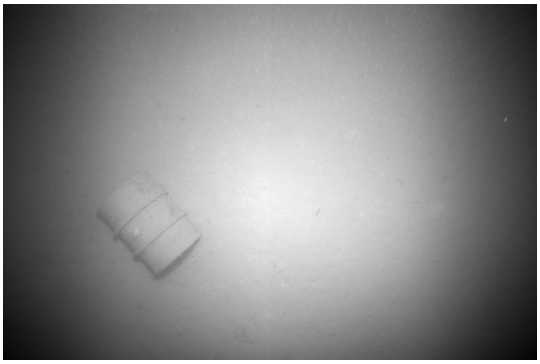
Teksturer

- Førsteordens statistikk
- GLCM

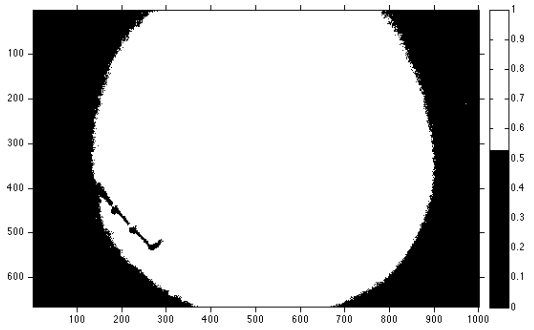
7.1 Skyggedeteksjon fra terskling

I bilder som inneholder objekter som stikker opp fra havbunnen, kan vi som tidligere nevnt på grunn av hjelpebelysningen forvente å finne skygger. Skygger kjennetegnes ved å være mørkere enn omgivelsene, så en fremgangsmåte kan være å terskle bildet slik at bare de mørkeste partiene segmenteres ut. Å terskle et bilde vil si å tilordne et piksel en ny verdi avhengig av om dets opprinnelige gråtoneverdi var over eller under en valgt grense. Vanligvis farges et piksel hvitt hvis opprinnelig verdi er over terskelen, og til svart ellers. *Global terskling* betyr å bruke en felles terskelverdi for alle piksler i bildet. Alternativet til dette er *lokal terskling* som bruker ulike terskler i forskjellige områder av bildet. Når mange bilder skal terskles, er det ønskelig å bestemme terskelverdien automatisk. *Lokalt adaptiv terskling* regner ut terskelen for et piksel basert på informasjon samlet fra et naboskap eller et vindu av en bestemt størrelse omkring det aktuelle pikselet. En sammenligning av flere metoder for to-nivå terskling er gjort i [39] og metodene jeg har testet er omtalt i [2] i tillegg til i [39]. Jeg kommer tilbake til adaptiv terskling i avsnitt 7.1.5.

Et undervannsbilde tatt med HUGIN AUV er vist i figur 7.1(a). Bildet viser et oljefat som har ligget en stund på mudderbunn. Et nytt lag med mudder er i ferd med å legge seg over oljefatet, så overflaten går veldig i ett med havbunnen. På grunn av lyskildens plassering i forhold til kameraet kan vi se hvordan skyggene i hovedsak dannes i bakkant av strukturene på oljefatet. Foreløpig er oljefatet helt og fint, så konturene er rette og sammenhengende og vi kan tydelig se de karakteristiske tverrbåndene. I bildet er det få andre elementer som virker forstyrrende. Det er lite fremtredende marin snø, og havbunnen har jevn kontrast. Samtidig kan vi kjenne igjen flere av de kjente problemene med undervannsbilder fra avsnitt 3.1. Et karakteristisk «lysende slør» ligger over bildet som følge av bakoverspredning, og foroverspredningen gjør bildet litt uskarpt. Tydeligst er den ujevne belysningen knyttet til bruken av kunstig lyskilde. Bildet er lysest i midten og blir gradvis mørkere mot kantene, helt til det grenser til svart i hjørnene.



(a) *Optisk kamerabilde tatt med HUGIN AUV.*



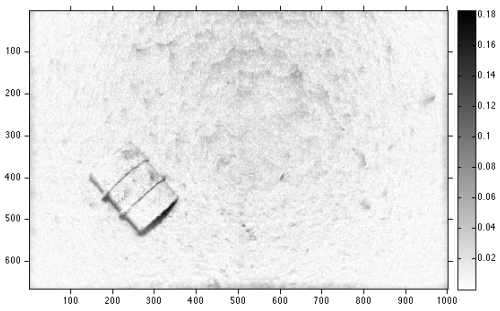
(b) *Global terskling av (a), med grense på 0.47.*

Figur 7.1: *Optisk undervannsbilde før og etter global terskling.*

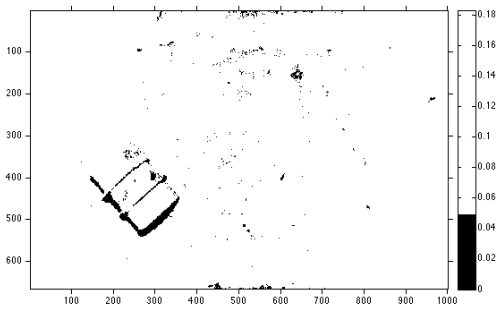
7.1.1 Global terskling og bakgrunnsutjevning

Ved inspeksjon av figur 7.1(a) er det tydelig at den ujevne belysningen vil gjøre skyggedeteksjon vanskelig, da store deler av bakgrunnen faktisk er mørkere enn skyggen under oljefatet. En global terskling av dette bildet vil føre til at det segmenteres ut en stor ring i ytterkant av bildet sammen med det skyggepartiet vi ønsker å detektere. Figur 7.1(b) viser resultatet av global terskling. Bildets gråtoneverdier spenner fra 0 til 1, og for dette bildet var 0.47 den høyeste terskelen som kunne settes uten at den ytre ringen trakk seg over objektet. Likevel har nesten ingenting av skyggen på kortsiden til oljefatet blitt segmentert ut, så skyggen er åpenbart lysere enn kantene av bildet.

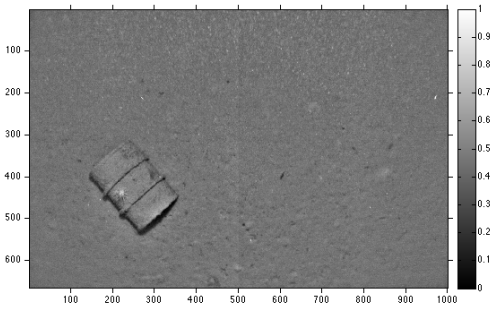
Det finnes flere teknikker for å kompensere for ujevn bakgrunnsbelysning. Jeg har testet to stykker, bottom-hat transform og homomorf filtrering. Disse er beskrevet i avsnittene 6.1 og 6.2. Bruk av disse er dokumentert i figur 7.2, og som resultatene viser har det i alle tilfeller skjedd stor forbedring fra terskling av det ufiltrerte bildet i figur 7.1(b). Terskelen er satt manuelt i disse eksemplene, med hensyn på å få sammenhengende kanter på objektet uten å inkludere for mye støy ellers i bildet. Bottom-hat gir grunnlag for god segmentering av objektet, men det oppstår noe struktur på havbunnen. Denne strukturen oppstår som følge av små partikler i bildet, som omtales nærmere i avsnitt 7.1.2. Et sirkelformet strukturelement med radius 15 eller 21 piksler har fungert greit for testbildene. Da oppnås god skyggedekteksjon uten at det blir for mye ringeffekter i bakgrunnen. Homomorf filtrering fungerer også godt. Siden dette i praksis er en høypassfiltering av bildet får vi høyere kontrast rundt kantene til objektet, men samtidig vil det også føre til at finkornet støy i bildet forsterkes. En kombinasjon av de to filtreringsteknikkene har også blitt testet ut, og resultat ble en slags mellomting av de to separate resultatene. Argumentasjonen for å teste dette var at en bottom-hat transform av et bilde med høyere kontrast også ville gi et filtrert resultat med høyere kontrast mellom forgrunn og bakgrunn. Bottom-hat transform vil i det tilfellet benyttes som en skyggedetektor, se avsnitt 6.1.



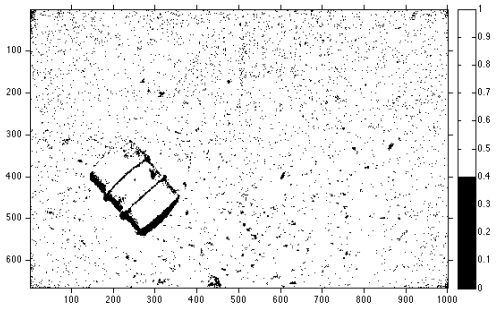
(a) *Bottom-hat transform* `strel('disk',21)`. *Merk invertert gråtoneskala.*



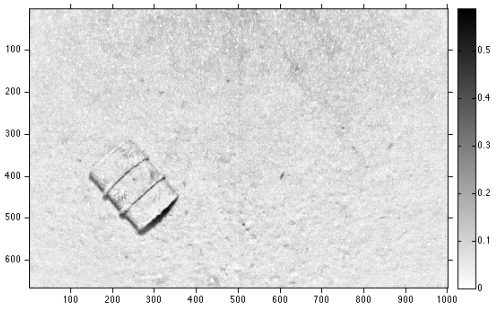
(b) *Global terskling av (a) med grense på 0.05.*



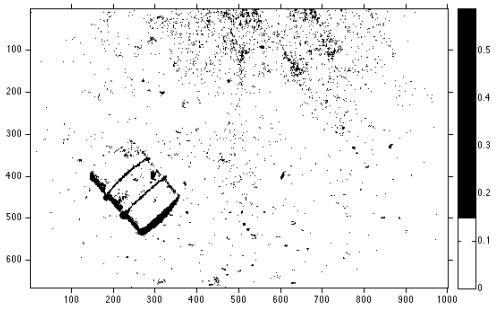
(c) *Homomorf filtrering (order = 1, cutoff = 0.01)*



(d) *Global terskling av (c) med grense på 0.40*



(e) *Først homomorf filtrering og deretter bottom-hat* `strel('disk',15)`.



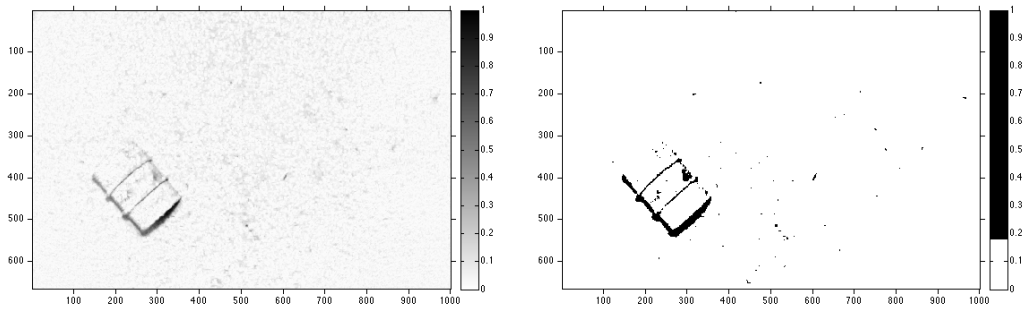
(f) *Global terskling av (e) med grense på 0.15.*

Figur 7.2: *Global terskling av bilder etter bakgrunnsutjevning.*

7.1.2 Filtrering av uønsket signal

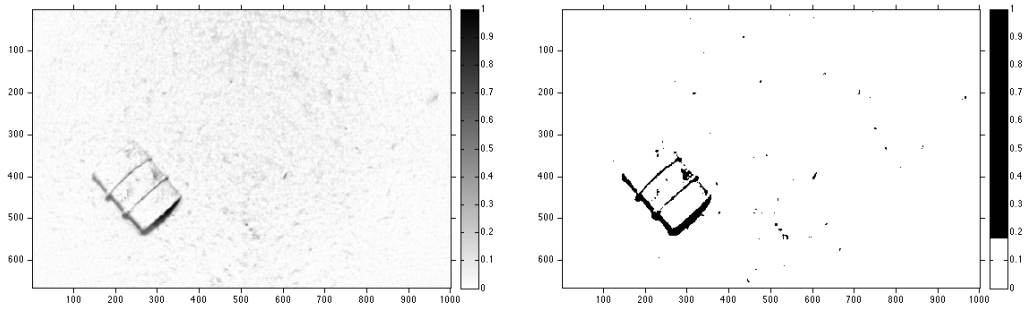
For å redusere finkornet støy i det tersklede bildet, dvs. å få færre forekomster av de små segmenterte områdene som ikke hører til objektet, har det blitt forsøkt med en støyfiltrering i forkant av bakgrunnsutjevningen og tersklingen. Støykilden er i all hovedsak marin snø, det vil si organiske partikler som blir avbildet som små, lyse prikker i bildet. To alternative teknikker som kan brukes mot denne typen støy er medianfiltrering og morfologisk åpning. Fordelen med medianfiltrering kontra morforlogisk åpning er at median vil fjerne både lyse og mørke outliers, såkalt «salt og pepper-støy», mens morforlogisk åpning bare vil dempe lyse felter, eller «salt». Denne forskjellen er mest aktuell i forbindelse med homomorf filtrering, som er et høypassfilter. Et høypassfilter vil forsterke finkornet støy, så det er ønskelig å fjerne så mye som mulig av denne, både lys og mørk. Filterstørrelsen bør samsvare med støyen som skal fjernes, og partiklene som ønskes fjernet her er relativt små. Jeg har testet filtrering med `MATLAB`-funksjonene `medfilt2` og `imopen` for henholdsvis medianfiltrering og morfologisk åpning, og eksperimentert litt med filterstørrelser. En vindusstørrelse på `[5 5]` for medianfiltreringen og et strukturelement `strel('disk',3)` for åpningen fungerte greit.

Eksempler fra eksperimenteringen er vist i figurene 7.3 og 7.4. I de tersklede bildene er terskelen satt manuelt slik at objektet skal segmenteres omtrent likt som i tidligere tersklingsforsøk uten støyfiltrering. Ut fra dette kan man si at støyfiltreringen gir resultater med tydeligere kanter i forhold til mengden støy. Det meste av bidraget fra marin snø har blitt fjernet, mens enkelte krepsehull på havbunnen med noe større diameter fortsatt er synlige. En morfologisk lukking ville kunnet fjerne mørke støyelementer fra bildet før terskling, men da med fare for å også fjerne smale skyggepartier som hører til objektet.



(a) *Morfologisk åpning før bottom-hat transform.*

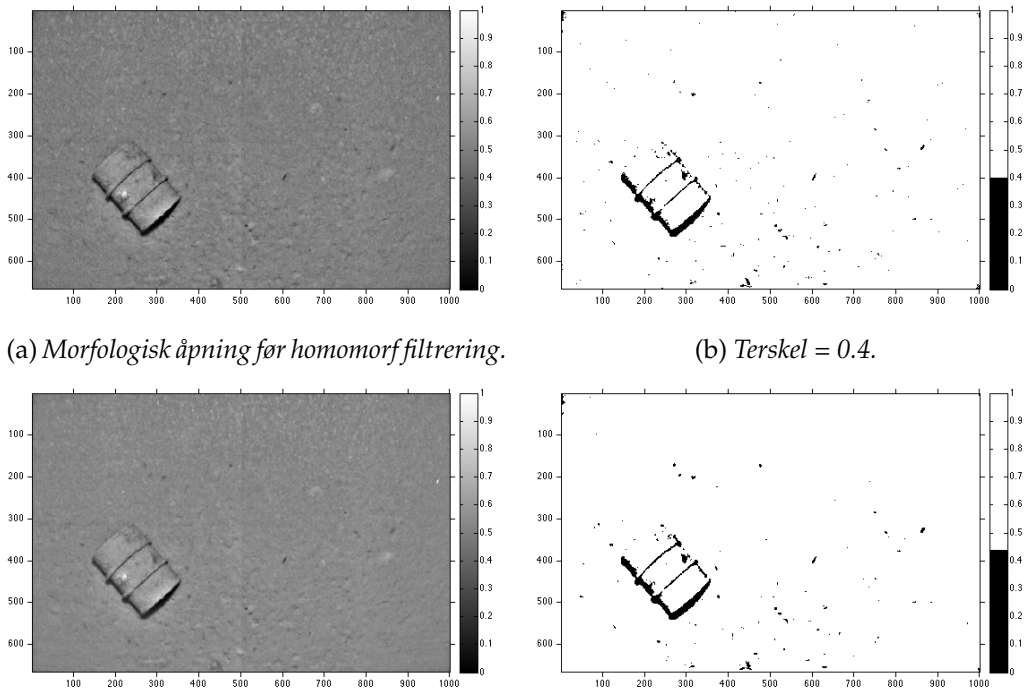
(b) *Terskel = 0.19.*



(c) *Medianfiltrering før bottom-hat transform.*

(d) *Terskel = 0.19.*

Figur 7.3: *Støyfiltrering før bakgrunnsutjevning med bottom-hat transform og global terskling. Morfologisk åpning og medianfiltrering har blitt prøvd. Strukturelement for åpning er* `strel('disk',3)` *og vindusstørrelse for medianfiltrering er 5x5. For bottom-hat er strukturelement* `strel('disk',21)`.



(a) *Morfologisk åpning før homomorf filtrering.*

(b) *Terskel = 0.4.*

(c) *Medianfiltrering før homomorf filtrering.*

(d) *Terskel = 0.45.*

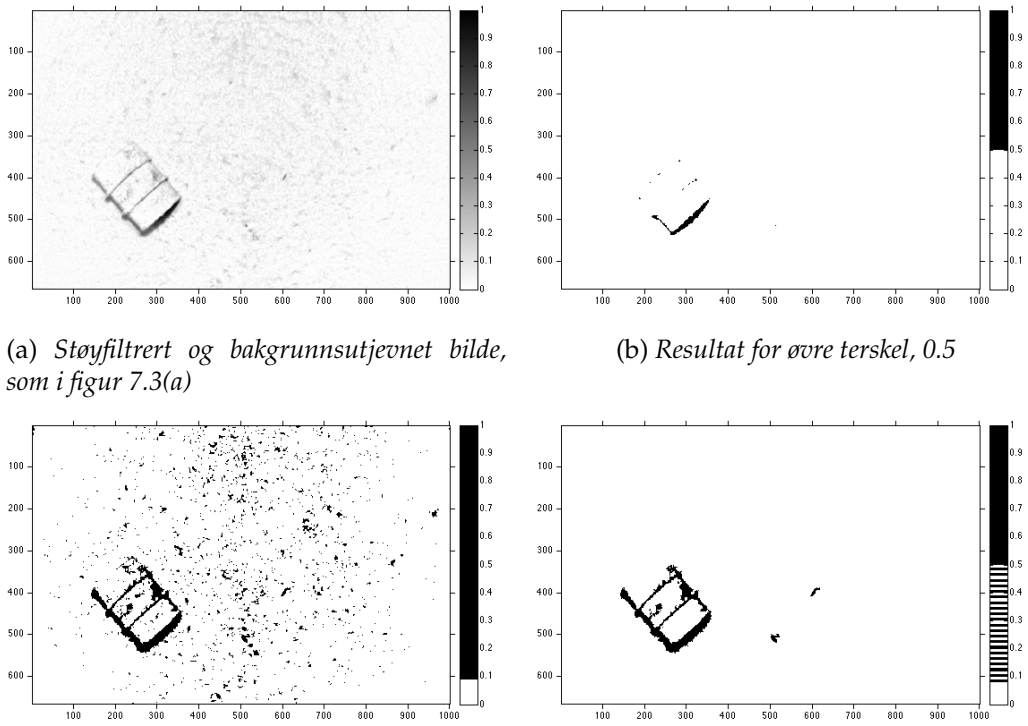
Figur 7.4: *Støyfiltrering før bakgrunnsutjevning med homomorf filtrering og global terskling. Morfologisk åpning og medianfiltrering har blitt prøvd. Strukturelement for åpning er* `strel('disk',3)` *og vindusstørrelse for medianfiltrering er 5x5. Parametere for homomorf filtrering er* `order = 1, cutOff = 0.01`.

7.1.3 Hysteresetersкling

Som et alternativ til den globale tersklingen i forrige avsnitt, har det også blitt prøvd hysteresetersкling. Terskling med hysterese består av to terskler, en øvre og en nedre. Ved å anta at egenskapene man leter etter finnes i form av større sammenhengende områder med noe varierende gråtoneverdi, setter man først en øvre grense som velges slik at så lite støy og så få uønskede elementer som mulig kommer over terskelen. Samtidig er det viktig å i det minste få med deler av de elementene man leter etter. Deretter settes en ny terskel som er lav nok til å segmentere ut alle verdier som hører til egenskapene vi leter etter, og som dermed også tillater mer støy. Fra denne siste tersklingen inkluderer man bare komponenter som henger sammen med komponenter segmentert ut ved øvre terskel. Med andre ord kan man terskle bildet med en relativt lav terskel, men man er sikker på at hver segmenterte region inneholder minst ett piksel som har høyere verdi enn øvre terskel, og på denne måten får man forhåpentligvis luket bort endel uønskede elementer. For segmentering av skygger må man enten invertere bildet slik at skygger vises som lyse partier i bildet, eller så endrer man bare reglene for øvre og nedre terskel. Et eksempel er vist i figur 7.5, der både resultat etter øvre og nedre terskling er vist, sammen med den endelige hysteresetersklingen. Her kan virkningen av hysteresetersкling sies å være bedre enn støyfiltrering og vanlig global terskling som omtalt i forrige avsnitt.

7.1.4 Otsus metode

Otsus metode [15] er en mye brukt teknikk for å finne gode terskelverdier. Ideen til metoden er å anta at klassene i bildet (forgrunn og bakgrunn, eller i tilfellet til denne oppgaven, kanter og ikke kanter) skiller seg fra hverandre i intensitetsverdi. For å separere klassene bruker Otsus metode gråtonehistogrammet til bildet som skal terskles. Det antas at det finnes en topp for forgrunnspiksler (høye verdier) og en topp for bakgrunnspiksler (lave verdier). Metoden leter etter punktet som skiller disse to toppene best mulig. Figur 7.6(a) viser et histogram der klassene er tydelig samlet i to separate topper. For undervannsbildene i denne oppgaven er derimot mengden forgrunnspiksler relativt liten i forhold til bakgrunnen, så histogrammet blir mer unimodalt, se figur 7.6(b). For at Otsus metode skal fungere godt bør størrelsesforholdet mellom toppene til forgrunn og bakgrunn ikke være større enn 1:10 [3], noe som ikke kan sies å være oppfylt i dette tilfellet. Metoden har derfor ikke blitt vurdert for global terskling av undervannsbildene, men har blitt testet kort i forbindelse med terskling av gradientmagnitude, se avsnitt 7.2.



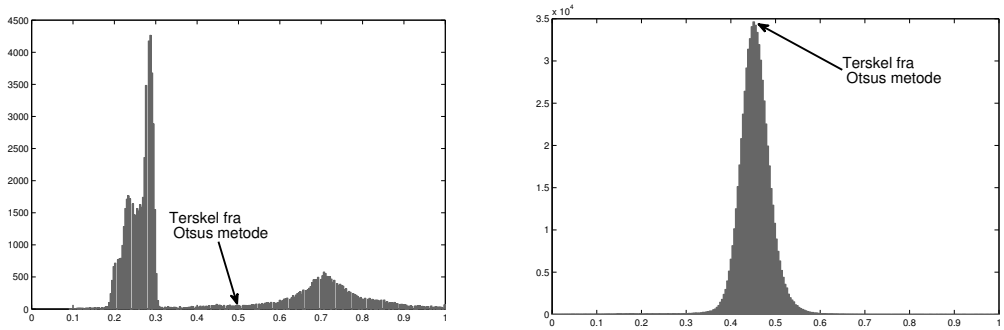
(a) *Støyfiltrert og bakgrunnsutjevnet bilde, som i figur 7.3(a)*

(b) *Resultat for øvre terskel, 0.5*

(c) *Resultat for nedre terskel, 0.1*

(d) *Resultat for hysteresetersklingen [0.5, 0.1], 8-naboskap.*

Figur 7.5: *Hysteresetersklingen. Komponenter fra segmentering med laveste terskel som også henger sammen med komponenter fra segmentering med øvre terskel blir med i det hysteresetersklede bildet. Bildene er invertert for bedre visning på papir.*



(a) *Bimodalt histogram fra bilde med godt separerte klasser.*

(b) *Unimodalt histogram fra homomorf-filtrert undervannsbilde.*

Figur 7.6: *Bimodalt (a) og unimodalt (b) histogram. Terskel fra Otsus metode er vist.*

7.1.5 Lokalt adaptiv terskling

Det har blitt testet to metoder for lokalt adaptiv terskling, *Bernsens metode* [2] og *Niblacks metode* [2]. For Niblacks metode har det blitt gjort forsøk på automatisk valg av parameteren k, med inspirasjon fra Yanowitz og Bruckstein metode [48].

Bernsens metode

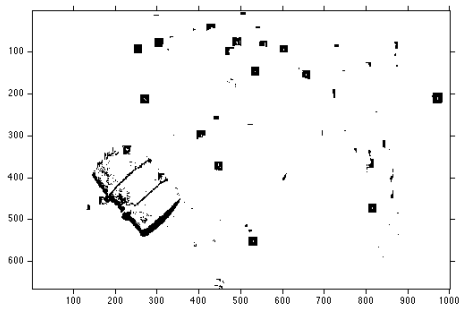
I Bernsens metode ser man på høyeste og laveste gråtoneverdi innenfor et vindu omkring hvert piksel, og bruker differansen mellom disse to til å avgjøre videre handlinger. Hvis forskjellen mellom høyeste og laveste verdi er større en en bestemt grense C antas vinduet å inneholde to klasser (forgrunn og bakgrunn), og pikselet klassifiseres i forhold til gjennomsnittet av de to verdiene. Hvis forskjellen er lavere enn grensen antas vinduet å inneholde bare en klasse. Slike unimodale tilfeller kan for eksempel håndteres ved å sette pikselet til forgrunn dersom dets verdi ligger i øvre halvdel av gråtoneskalaen, og til bakgrunn i motsatt tilfelle. Jeg har valgt å klassifisere det som bakgrunn i alle tilfeller der kontrasten i vinduet er for lav. Beslutningsregelen i sin helhet er vist i likning (7.1), hvor $t(x, y)$ er terskelen for et enkelt piksel, Z_{high} og Z_{low} er hhv. høyeste og laveste gråtoneverdi i vinduet og C er kontrastgrensen.

$$t(x,y) = \begin{cases} \frac{Z_{high}+Z_{low}}{2} & \text{if } Z_{high} - Z_{low} > C \\ 1 & \text{else} \end{cases} \tag{7.1}$$

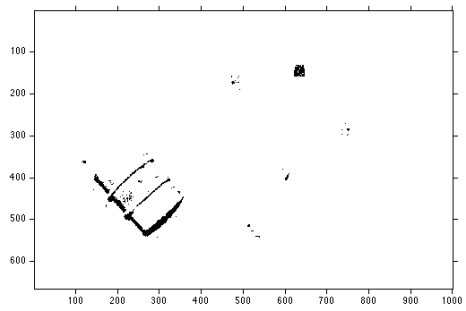
Støy i bildet, dvs. enkeltpiksler med svært avvikende verdi, påvirker utregningen av terskelverdi. Rundt slike piksler oppstår det områder tilsvarende valgt vindusstørrelse der kontrasten tilsynelatende er høy, med feilaktig utregning av terskelverdi og videre feilaktig klassifisering som resultat. En løsning er å ikke velge høyeste og laveste pikselverdier som grenser, men plukke verdier fra bestemte persentiler av histogrammet eller på annen måte undertrykke outliers.

En annen ulempe er at dette er en vindusbasert metode, så prosesseringstiden blir relativt lang for hvert bilde. På min PC har jeg fått tiden ned i ca 10 sekunder ved bruk av `colfilt`-funksjonen i `MATLAB`.

To bilder som har blitt tersklet med Bernsens metode er vist i figur 7.7. Parameteren C og vindusstørrelse har blitt satt manuelt for nær optimal segmentering av skygger. Det er ikke tatt hensyn til outliers, så vi kan se at det oppstår endel blokk-effekter som følge av bimodalitet (det vil i dette tilfellet si stor forskjell mellom lyseste og mørkeste gråtone), særlig i det ufiltrerte bildet i figur 7.7(a).



(a) *Terskling av ufiltrert bilde.*
C = 0.1, vindu = [20 x 20]



(b) *Terskling av bilde etter bottom-hat og homomorf filtrering.*
C = 0.5, vindu = [31 x 31]

Figur 7.7: *Terskling med Bernsens metode. Metoden fungerer relativt godt på det ufiltrerte bildet i (a), men vi ser at marin snø fører til endel feilklassifiseringer. Etter bakgrunnsutjevning i (b) er ikke dette problemet like fremtredende. Dette kan skyldes økt vindusstørrelse, men hovedårsaken er antagelig økt kontrast i kildebildet. Legg merke til at C har økt fra 0.1 i (a) til 0.5 i (b). Kildebildene kan sees i figur 7.1(a) og figur 7.2(e).*

Niblacks metode

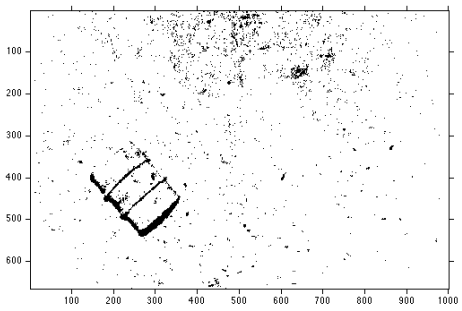
Ideen til denne metoden er å sette terskler som avhenger av statistisk informasjon lokalt i bildet. Klassen til pikselet regnes ut som vist i likning (7.2), hvor μ_{win} og σ_{win} er henholdsvis lokal middelverdi og standardavvik innenfor et vindu omkring et piksel. Parameteren k er antall standardavvik og justerer sensitiviteten til funksjonen.

$$g(x) = \begin{cases} 0 & \text{if } f(x) < \mu_{win} + k\sigma_{win} \\ 1 & \text{else} \end{cases} \tag{7.2}$$

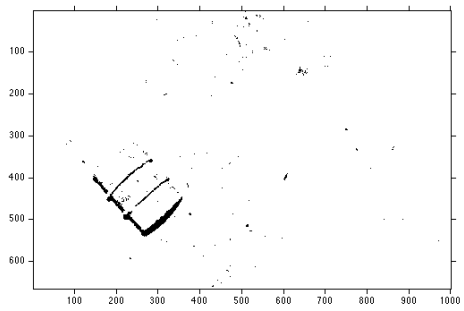
Selv om dette i utgangspunktet er en vindusbasert metode, er det ingenting i veien for å bruke den til global terskling. Da regner man ut middelverdi og standardavvik for hele bildet og bruker samme utregnede terskel for alle piksler. Dette kan være aktuelt for bilder der belysningen er jevnet ut, som ved for eksempel homomorf filtrering eller bottom-hat transform.

Som vindusbasert metode erfarte jeg at den krever relativt store vinduer. Ved bruk av små vinduer er det større sannsynlighet for at området som dekkes er homogent og dermed får et veldig lavt standardavvik. Dette gjør at terskelen ligger nær middelverdien i det aktuelle vinduet, og sannsynligheten for å feilklassifisere piksler vil være stor. Robustheten øker med større vindu, men det gjør også prosesseringstiden. Med vindusstørrelser opp mot [49 x 49] ligger prosesseringstid på over 60 sekunder per bilde på min PC.

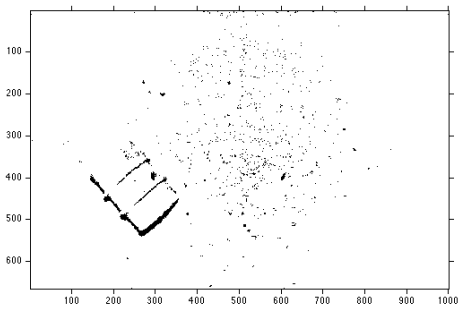
Automatisk valg av k Jeg har prøvd global terskling ved hjelp av Niblacks metode og automatisk valg av k. Dette ble gjort ved å teste ulike valg for k. Følgende verdier



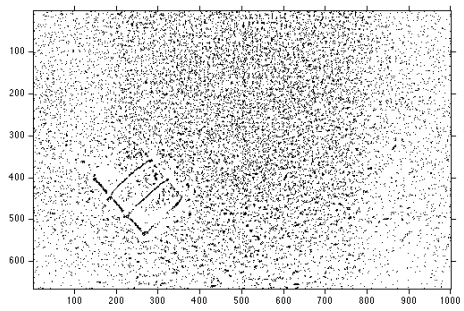
(a) *Global terskling med k = -1.8. Bildet er forprosessert med bottom hat og homomorf filtrering.*



(b) *Global terskling med automatisk valg av k: -3.0. Bildet er forprosessert med bottom hat og homomorf filtrering.*



(c) *Glidende vindu, [49 x 49], k = -1.0. Ingen forprosessering av bildet.*



(d) *Glidende vindu, [15 x 15], k = 0.75. Ingen forprosessering av bildet.*

Figur 7.8: *Resultater fra eksperimentering med Niblacks metode for ulike valg av parameterverdi* `k` *og vindusstørrelse, samt global og adaptiv terskling.*

ble testet: `k = [−4.5 −4 −3.5 −3.25 −3 −2.75 −2.5 −2 −1.5 −1]`. For hvert forsøk på terskling ble det laget et kantbilde `edges` av det tersklede bildet `bw` ved hjelp av `edge(bw,'sobel')`. Dette gir kantene til de segmenterte objektene (se avsnitt 7.2.1). For hver sammenhengende komponent av kanter (fra `bwconncomp(edges, 8)`) ble det regnet ut gjennomsnittlig gradientmagnitude. Dette ble gjort ved å se på magnituden i gradientbildet i samme pikselposisjoner som punktene langs komponentens kant. Høyeste gjennomsnittlige gradient ble sammenlignet i kantbildene (fra ulike k-verdier), og den k-verdi som var opphav til høyeste gjennomsnittlige gradient ble valgt. Dette er inspirert av postprosesseringen i Yanowitz og Brucksteins metode [48], der en lignende teknikk brukes for å fjerne falske objekter etter terskling.

7.1.6 Diskusjon av tersklingsresultater

I de foregående avsnitt har det blitt testet ulike metoder for å detektere objekter på havbunnen ved hjelp av skyggesegmentering. En kort oppsummering av resultatene er presentert under:

Global terskling
Uaktuelt med tanke på videre bruk, grunnet ujevn belysning. Metoden er vist som argumentasjon for utprøving av andre alternativer.

Global terskling og bakgrunnsutjevning
OK resultater, men bottom-hat skaper noen falske objekter/strukturer på havbunnen, og med homomorf filtrering blir høyfrekvente bidrag mer fremtredende.

Filtrering av uønsket signal
Morfologisk åpning og medianfiltrering har blitt testet. Begge metoder fungerer etter hensikten, men medianfiltrering er vurdert som mer allsidig.

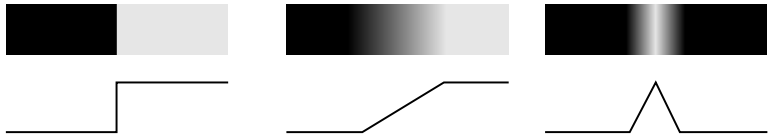
Hysteresetterskling
God segmentering av kanter uten å få med de uønskede strukturene på havbunnen. Foretrukken metode.

Otsus metode
Ikke vurdert, da kriteriene for å kunne bruke metoden ikke synes å være oppfylt.

Bernsens metode
Fungerer relativt greit, men kan skape noen blokkeffekter rundt forekomster av marin snø i bildet. Problemet reduseres med økt vindusstørrelse, og kan antagelig reduseres med støyfiltrering.

Niblacks metode
Mindre vellykket enn Bernsens metode med tanke på sensitivitet for støy. Interessant å gjøre forsøk med automatiske terskelvalg, som syntes å fungere godt.

Dersom det skulle velges én metode å gå videre med, er det hysteresetterskling som har fremstått som den mest robuste teknikken. Hvis det i tillegg skulle velges en metode for støyfiltrering, vil medianfiltrering være det foretrukne valget.



(a) *Steg-kant* (step edge) (b) *Rampe* (ramp edge) (c) *Tak-kant* (roof edge)

Figur 7.9: *Forskjellige typer kanter. Illustrasjon fritt etter [15].*

7.2 Kanter/gradienter

I dette avsnittet har det blitt forsøkt å finne objekter ved hjelp av kantdeteksjon, det vil i første omgang si å finne en kantdeteksjonsalgoritme som fungerer bra på undervannsbilder. Kantbildene må prosesseres videre med for eksempel terskling av gradientmagnitude og deretter en houghtransform for å lokalisere rette linjer i bildene. Bakgrunnstoffet til avsnittet er i hovedsak hentet fra [15, 24].

For å detektere punkter, linjer og kanter i bilder brukes det ofte metoder som ser på endring i intensitet mellom nabopiksler. Denne endringen kan være brå eller gradvis, illustrert som henholdsvis steg-kant og rampe i figur 7.9. Den tredje varianten er en såkalt tak-kant, som er typisk for punkter og linjer. Disse består egentlig av to kanter, først en overgang fra mørkt til lyst og umiddelbart etter en overgang fra lyst til mørkt. Intensitetsendringer i bildet kan detekteres ved hjelp av numerisk derviasjon. Den deriverte vil bli 0 i områder uten endring i intensitet, og den vil få en positiv eller negativ verdi der det finnes en endring. For å finne både styrken og retningen til en kant, kan vi regne ut gradientvektorer for alle piksler i bildet. Gradientvektoren for et pikselkoordinat er sammensatt av punktets deriverte i både x og y-retning, det vil si den partiellderiverte med hensyn på x og y:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} g_x \\ g_y \end{bmatrix} \tag{7.3}$$

Magnituden mag til denne vektoren indikerer styrken (tydeligheten) til kanten, mens fasen α gir kantens retning.

$$mag(\nabla f) = \sqrt{g_x^2 + g_y^2}, \qquad \alpha(\nabla f) = tan^{-1}\left[\frac{g_y}{g_x}\right] \tag{7.4}$$

7.2.1 Sobel gradientoperator

For å få en tilnærming til de to komponentene i gradientvektoren, kan det brukes ulike konvolusjonsfiltere eller «gradient-operatorer». I oppgaven har jeg først testet Sobel-operatoren [47]. Filterkjernen er et produkt av en deriverende kjerne (som finner endringer i gråtoneverdier) og en midlende filterkjerne. Sobelfilteret trekker dermed

ut kanter samtidig som det utfører en glatting. De to gradientoperatorene i et 3x3 Sobelfilter er vist i likning (7.5):

$$h_y(x,y) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad h_x(x,y) = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{7.5}$$

Gradientoperatoren kan utføre kraftigere glatting ved å først konvolvere den med en annen glattende kjerne. Dette vil gi en større gradientoperator.

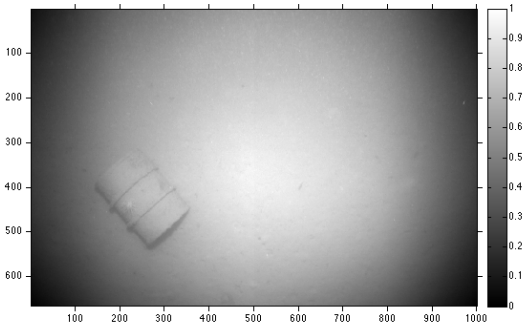
$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 2 & 8 & 12 & 8 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & -8 & -12 & -8 & -2 \\ -1 & -4 & -6 & -4 & -1 \end{bmatrix}$$

Ved å gjenta denne konvolusjonen kan man iterativt generere større og større gradientoperatorer som gir kraftigere og kraftigere glatting.

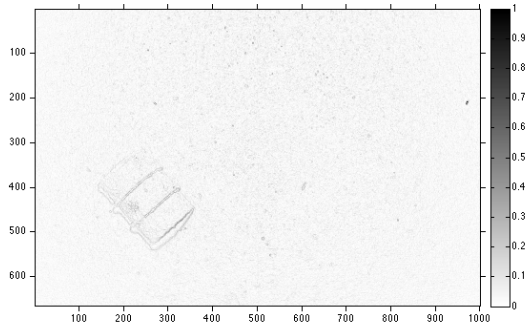
```
g = [1 2 1]' * [1 2 1];
sob3x3 = fspecial('sobel');
sob5x5 = conv2(g, sob3x3);
sob7x7 = conv2(g, sob5x5);
...
```

I figur 7.10 vises filtrering av et undervannsbilde med en 3x3 Sobel-operator. Figur (b) og (c) viser gradientmagnitude og gradientvinkel som gitt av likning (7.4). Til utregning av α brukes atan2-funksjonen for å få korrekt vinkel. Magnitudebildet er forsøkt tersklet for å trekke ut de tydeligste og mest interessante kantene, vist i figur (d). Som ved skyggedeteksjon blir partikler i vannet og strukturer på havbunnen også her segmentert ut som forgrunnsobjekter. Ved å bruke en større filterkjerne som glatter mer, vil disse små elementene smøres ut og få mindre tydelige kanter. De kraftige kantene rundt objektet vil i større grad bevares, selv om også disse vil smøres ut og bli tykkere. Resultatet av en slik filtrering er vist i figur 7.11, der det er brukt en 15x15 Sobeloperator. En konklusjon er at jo større filterkjernen er, jo bedre blir støyfiltreringen. Samtidig smøres kantene ut og blir tykkere, så et for stort strukturelement er heller ikke ønskelig. Ulike størrelser har blitt testet under eksperimenteringen, og 15x15 ble vurdert til å være den beste størrelsen.

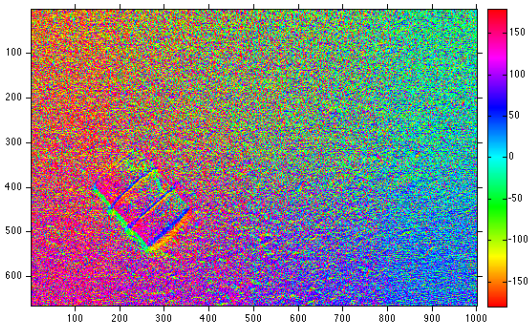
Ved nærmere inspeksjon av bildene i figur 7.10 kan vi se at kantene omkring oljefatet gir dobbel respons, eller doble kanter, i magnitudebildet, fordi dette er tak-kanter som beskrevet tidligere. Det kan forøvrig også legges merke til at den ujevne belysningen i originalbildet fører til gradientvektorer som har retning ut fra sentrum. Dette kan sees i figur 7.10(c) som viser gradientretningen. Ved å følge kanten rundt bildet ser vi at den generelle retningen varierer gradvis i takt med fargeskalaen fra -180 til 180 grader. Dette vil kunne påvirke retningen til faktiske kanter på objekter i bildene. Effekten av ujevn belysning kommer ikke like tydelig frem i magnitudebildet, fordi denne variasjonen i belysning gir en svært langsomt stigende rampe-kant og dermed svært lave magnitudeverdier.



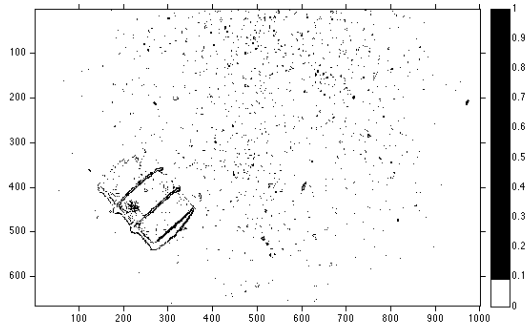
(a) *Originalbilde*



(b) *Gradientmagnitude*

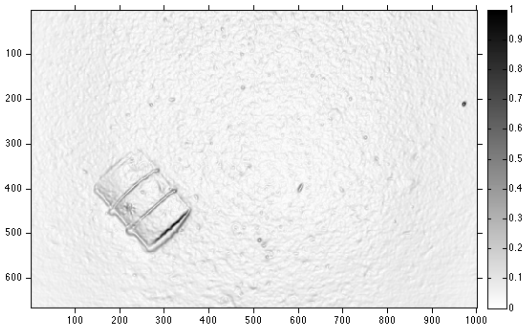


(c) *Gradientretning*

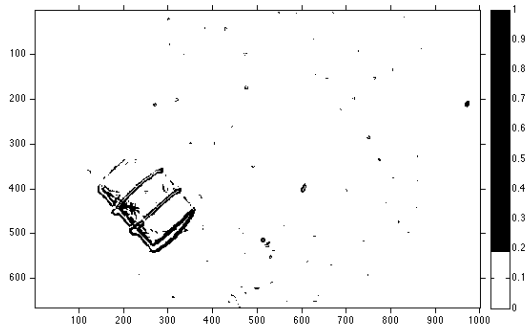


(d) *Global, manuell terskling av gradientmagnitude (0.1)*

Figur 7.10: *Sobelfiltrering, 3x3. Gradientmagnitude er normalisert til [0 1].*



(a) *Gradientmagnitude*



(b) *Manuelt tersklet gradientmagnitude (0.2)*

Figur 7.11: *Sobelfiltrering, 15x15. Gradientmagnitude er normalisert til [0 1].*

Effekten av homomorf filtrering

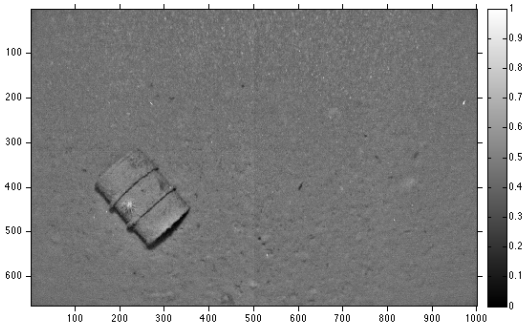
Som tidligere nevnt vil en homomorf filtrering medføre en høypassfiltrering av bildet. Dette er med på å forsterke kanter, og vil dermed kunne være fordelaktig før en kantdeteksjon. I tillegg vil filtreringen fremheve detaljer som ellers kan ligge skjult i hjørnene av bildet, så det finnes flere argumenter for å foreta en slik korreksjon. Ulempen er igjen at små, uønskede objekter vil bli mer tydelige. Resultatene av homomorf filtrering før kantdeteksjon med sobel er vist i figur 7.12. Ved en sammenligning av figur 7.10(b) og figur 7.12(b) kan vi se at det detekteres flere kanter på havbunnen etter homomorf filtrering, men også at kantene omkring oljefatet får høyere gradientmagnitude.

7.2.2 Kantbevarende glatting

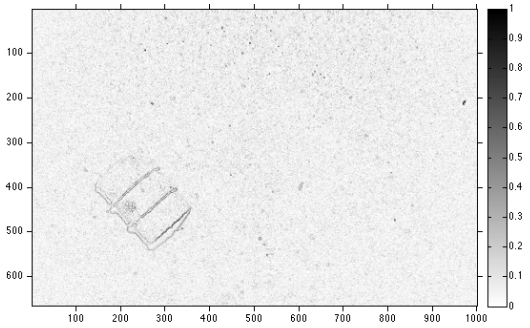
I avsnitt 7.2.1 ble Sobel-operatoren iterativt konvolvert med en annen kjerne som tilnærmet en Gauss-funksjon for å oppnå kraftigere glatting. Et alternativ til dette Gauss-filteret har vært å teste anisotrop diffusjon, beskrevet i avsnitt 6.3. Denne metoden utfører også glatting, men uten å dempe viktig kantinformasjon i bildet, noe som åpenbart vil være en fordel i forkant av kantdeteksjon. En ulempe med algoritmen er at den er avhengig av gode valg for både antall iterasjoner og sensitivitetsparameteren κ. I figurene 7.13 og 7.14 vises eksperimentering med manuell terskling av gradientmagnitude etter anisotrop diffusjon. Som figurene viser fjerner filtreringen mye av strukturene på havbunnen, og i segmenteringen oppnås relativt sammenhengende kanter. Samtidig kan vi, som påpekt i avsnitt 6.3, se at marin snø fremheves ved anisotrop diffusjon fordi kontrasten til havbunnen er for stor. At det blir med noen elementer i segmenteringen som ikke stammer fra objektets kanter er riktignok ikke kritisk, men det blir likevel vurdert da det sier noe om effekten av filtreringen. Fra resultatene av eksperimenteringen har dette blitt vurdert som et aktuelt filtreringsalternativ, men fordi effekten er såpass avhengig av parametervalg er ikke dette umiddelbart en foretrukken metode.

7.2.3 Cannys kantdetektor

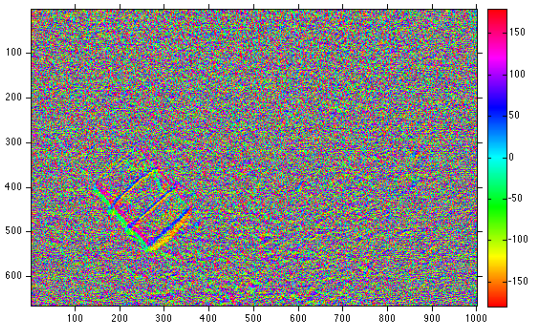
I avsnitt 7.1.3 ble det brukt hysteresetterskling for å få de detekterte skyggene til å bli mer sammenhengende. Dette er i praksis en form for regiongroing, der man kan tillate at segmenterte områder knyttes sammen dersom området mellom dem har visse egenskaper. I gradientbilder kan dette benyttes som kantlinking, der gradientmagnitude bestemmer om segmenterte kantelementer knyttes sammen eller ikke. Cannys kantdetektor [15] er en algoritme som tar i bruk slik regiongroing. Algoritmen er bygget opp av flere trinn, som til sammen skal sikre lav feilrate (få falske kanter), høy nøyaktighet (detekterte kanter skal ligge nær de faktiske kanter) og smal respons (kun ett punkt for hvert sanne kantpunkt). For å få til dette gjøres først en glatting av bildet med et Gauss-filter, før gradientmagnitude og retning regnes ut med en gradientoperator. Deretter tynnes kantene og til slutt utføres regiongroing



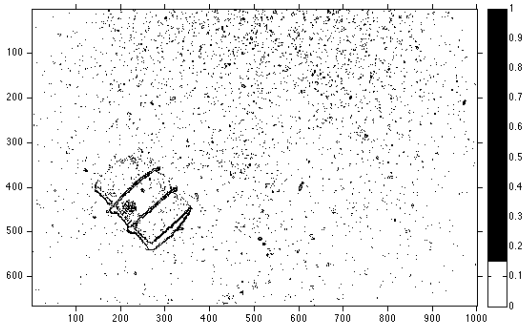
(a) *Originalbilde med homomorf filtrering*



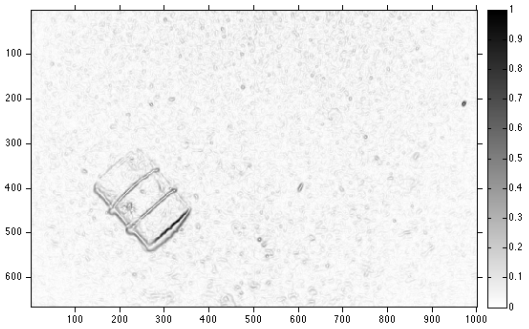
(b) *Gradientmagnitude normalisert til [0 1], 3x3 Sobel*



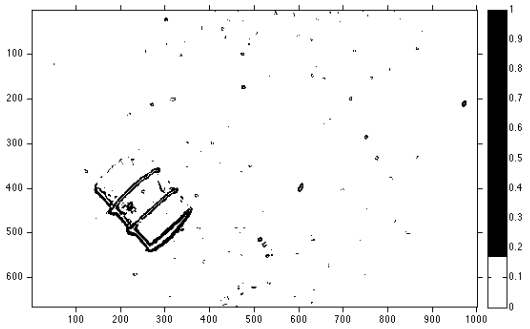
(c) *Gradientretning [-180, 180], 3x3 Sobel*



(d) *Manuelt tersklet gradientmagnitude, 3x3 Sobel. Terskel: 0.15*

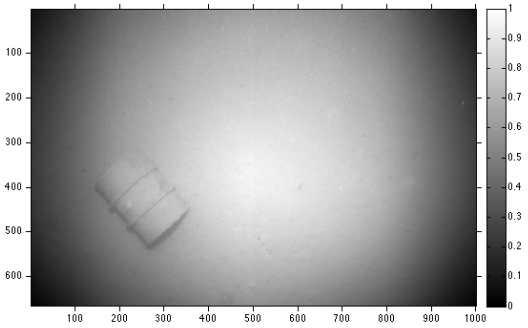


(e) *Gradientmagnitude normalisert til [0 1], 15x15 Sobel*

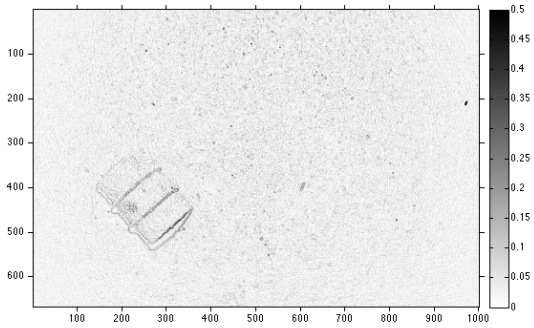


(f) *Manuelt tersklet gradientmagnitude, 15x15 Sobel. Terskel: 0.18*

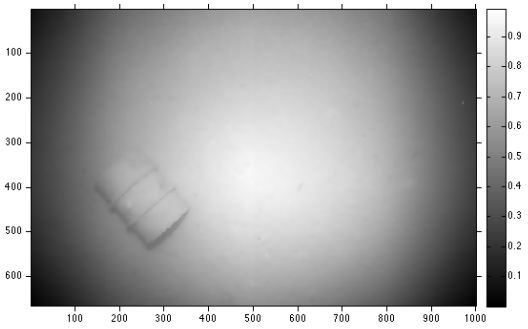
Figur 7.12: *Sobelfiltrering og terskling av bilde forprosessert med homomorf filtrering.*



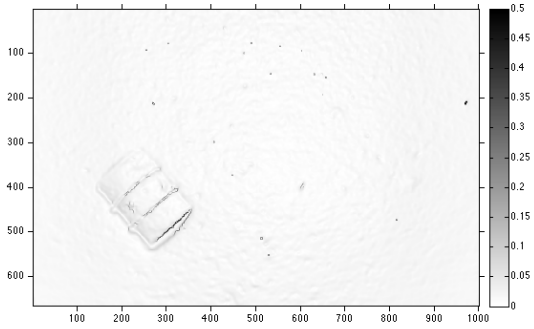
(a) *Ufiltrert undervannsbilde*



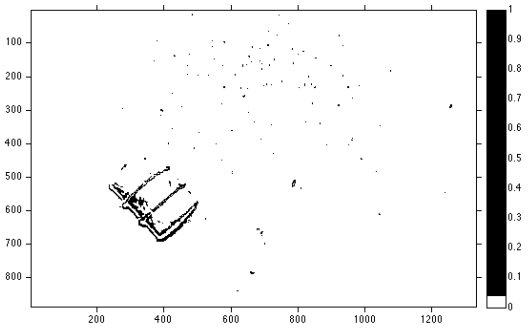
(b) *Gradientmagnitude, fra Sobel 3x3 gradient-operator*



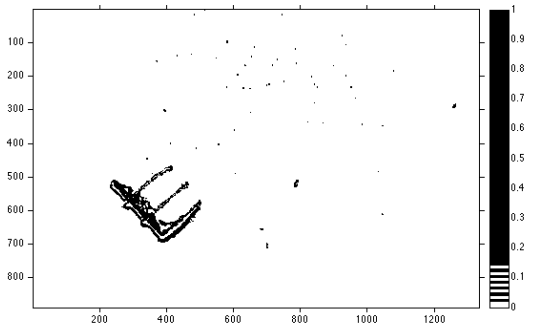
(c) *Anisotrop diffusjon, 20 iterasjoner,* $\kappa = 0.02, \lambda = 0.25$



(d) *Gradientmagnitude av filtrert bilde, fra Sobel 3x3 gradientoperator*

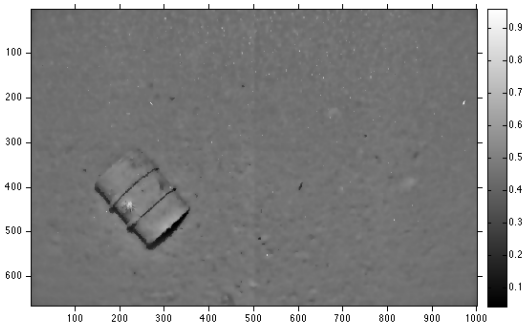


(e) *Manuell terskling av magnitudebilde, terskel: 0.04*

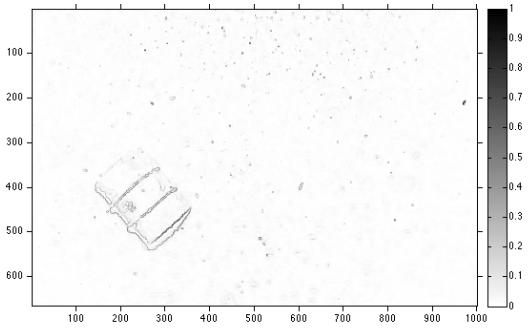


(f) *Manuell hysteresetерskling av magnitudebilde, terskler: 0.15 og 0.03*

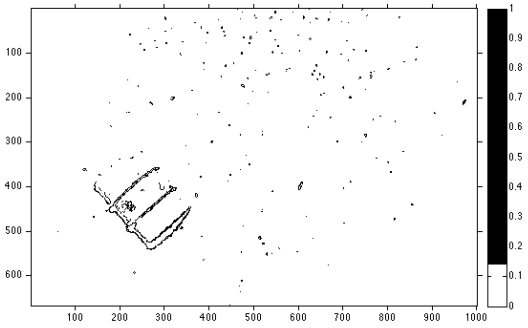
Figur 7.13: *Kantbevarende glatting med anisotrop diffusjon av ufiltrert undervannsbilde. Gradient-magnitude vist før og etter filtrering. For bedre visning på papir er gråtoneskalaen komprimert til [0, 0.5] i (b) og (d), slik at alle verdier fra 0.5 til 1 tegnes som svart.*



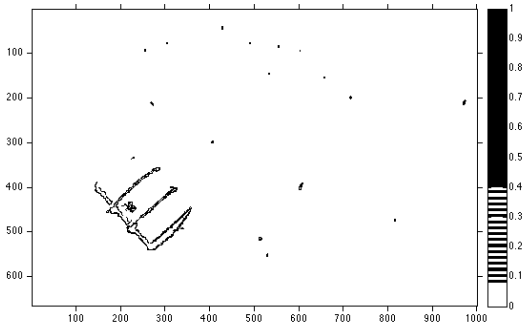
(a) *Kantbevarende glatting med anisotrop diffusjon. 20 iterasjoner,* $\kappa = 0.04, \lambda = 0.25$.



(b) *Gradientmagnitude av filtrert bilde, Sobel 3x3*

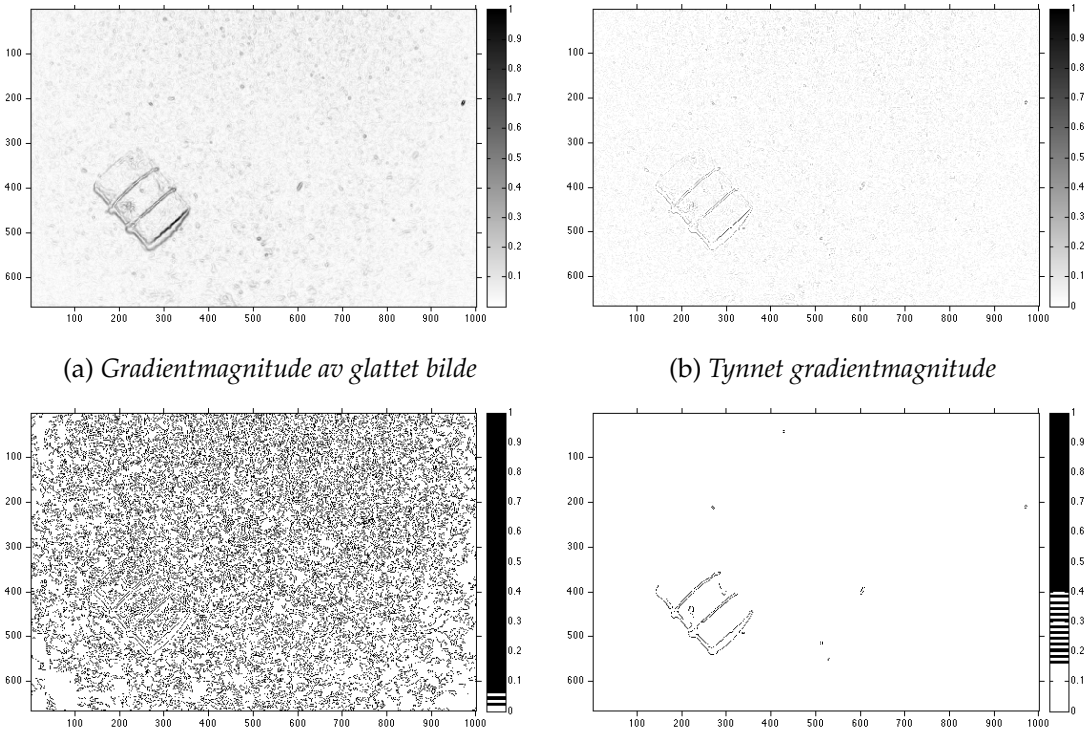


(c) *Manuell terskling av magnitudebilde, terskel: 0.14*



(d) *Manuell hysteresetерskling av magnitudebilde, terskler: 0.4 og 0.1*

Figur 7.14: *Anisotrop diffusjon av homomorf-filtrert undervannsbilde (vist tidligere i figur 7.12(a)).*



(a) *Gradientmagnitude av glattet bilde*

(b) *Tynnet gradientmagnitude*

(c) *Tersklet med* `MATLAB`*s automatiske parametervalg.*

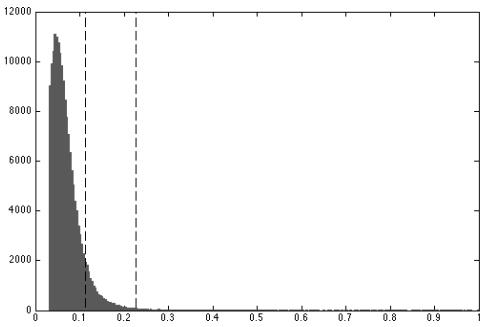
(d) *Tersklet ved 40% og 18%*

Figur 7.15: *Ulike trinn i Cannys algoritme for kantdeteksjon.*

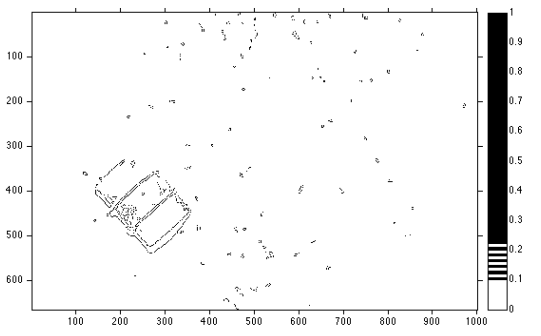
ved hjelp av hystereseterskling. Canny foreslår selv at forholdet mellom øvre og nedre hystereseterskel bør være 2:1 eller 3:1. En mer detaljert beskrivelse av Cannys kantdetektor finnes i [15].

Mulige parametere til `MATLAB`-implementasjonen av denne algoritmen er verdiene for hystereseterskling og standardavviket til Gauss-filteret. Figur 7.15 viser gradientmagnitude før og etter tynning, og to ulike valg av terskelverdier. Parameteren til Gaussfilteret er satt til $\sqrt{2}$, som er standard. Et forsøk på automatisk valg av terskel er vist i figur 7.16. Otsus metode er brukt på en vektor som inneholder alle verdier for gradientmagnitude større enn 0. Dette brukes som øvre hystereseverdi. Nedre verdi er satt til halvparten av øvre, altså et forhold på 2:1. Utregnede verdier for dette bildet ble henholdsvis 0.23 og 0.11.

De foreløpige eksperimentene har vist at Canny kan være en aktuell metode å gå videre med, da denne utfører både glatting, tynning og hystereseterskling, operasjoner som det i tidligere forsøk har vist seg å være et behov for. En utfordring er valg av terskelverdier, som omtales i neste avsnitt.



(a) *Histogram til gradientmagnitude > 0. Øvre og nedre utregnede terskel vist med stiplede linjer.*



(b) *Resultat etter terskling med utregnede verdier: 0.23 og 0.11.*

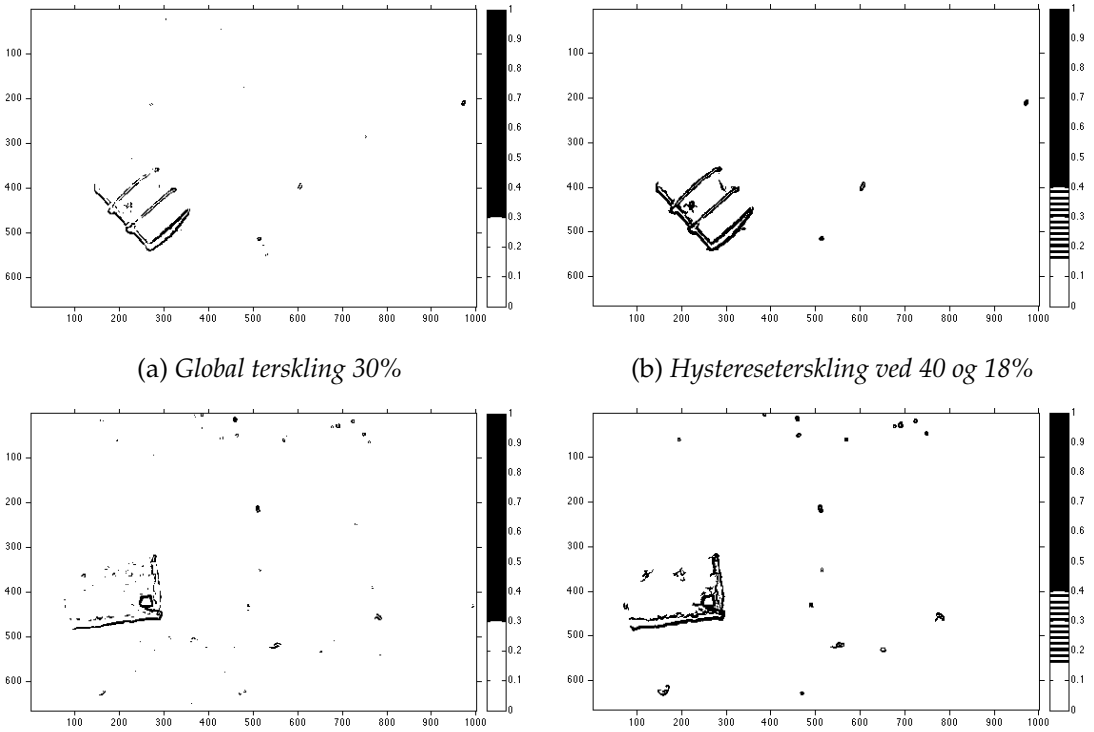
Figur 7.16: *Hysteresetersklер utregnet med Otsus metode. Forholdet mellom øvre og nedre terskel er 2:1.*

7.2.4 Valg av terskelverdi for gradientbilder

I datasett som består av flere bilder vil gradientmagnituden være relativt forskjellig fra bilde til bilde. Noen objekter har tydelige og skarpe kanter som gir høy magnitude, mens andre objekter kan gå mer i ett med havbunnen og få en slakere gradient med lavere magnitude. Dette medfører at en felles terskelverdi for alle bilder vil være utelukket, så informasjon fra hvert enkelt bilde må brukes til å velge individuelle verdier. En løsning som har blitt testet er å sette terskelen ved en prosentverdi av høyeste magnitude. Det er gjort forsøk med å normalisere magnituden til verdier fra 0 til 1 og deretter sette en terskel ved 0.3, altså at alle kanter med magnitude over 30% av høyeste magnitude defineres som kant. I tillegg ble det forøkt med hysteresetersкling med øvre grense lik 0.4 og nedre grense lik 0.18. Dette fungerte relativt greit på et lite subsett av bilder som ble brukt til testing i starten. Figur 7.17 viser to slike tilfeller.

Ved senere testing på andre bilder ble det raskt tydelig at dette er en lite robust metode, da den er svært ømfintlig for outliers. Problemer oppstår dersom objektene det letes etter har lav kontrast i forhold til havbunnen, samtidig som marin snø eller andre elementer avbildes som lyse felter med høy kontrast til havbunnen. Jo høyere kontrast, jo mer vil kanten ligne en steg-kant og magnituden blir følgelig høyere. En prosentverdi av magnituden til disse tilnærmede steg-kantene kan dermed ligge for høyt i forhold til kantene omkring objektene som skal detekteres. Et slikt tilfelle er illustrert i figur 7.18. Det er flere metoder som kan prøves for å håndtere problemet med outliers. I de følgende avsnitt omtales noen alternativer.

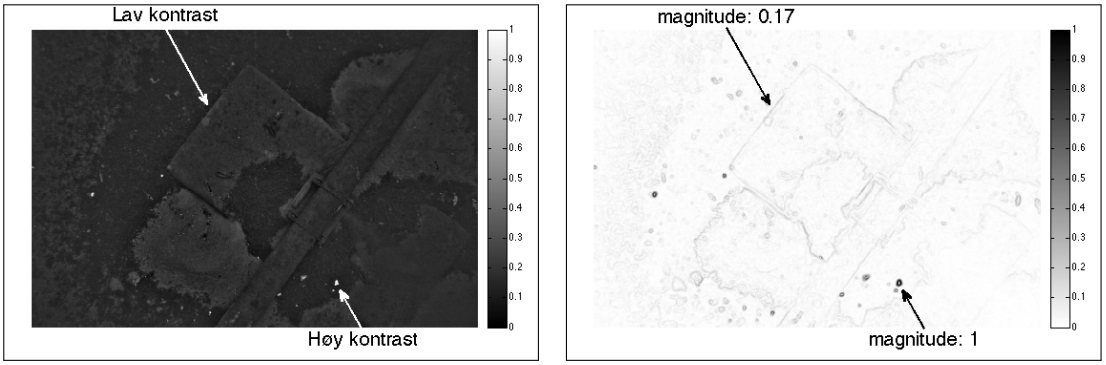
Fjerne outliers En løsning kan være å fjerne de uønskede objektene ved å modifisere det originale gråtonebildet, enten ved å tilordne de lyseste feltene en reservert «forbudt» verdi, eller ved å tilordne dem en lavere gråtoneverdi, for eksempel gjennomsnittlig gråtone fra et område omkring det aktuelle feltet. Å modifisere datagrunnlaget på denne måten er som regel ikke å foretrekke, da man risikerer å fjerne



(a) *Global terskling 30%* (b) *Hysteresetерskling ved 40 og 18%*

(c) *Global terskling 30%* (d) *Hysteresetерskling ved 40 og 18%*

Figur 7.17: *Eksempel på to bilder der terskelverdi satt som prosentvis verdi av høyeste magnitudeverdi fungerer greit. Det er brukt et 15x15 sobelfilter på bilder behandlet med homomorf filtrering. Originalbildet til (c) og (d) er vist i figur 4.1(d) på side 21.*



(a) *Homomorf-filtrert undervannsbilde* (b) *Gradientmagnitude med Sobel 15x15*

Figur 7.18: *En outlier, her i form av et lite objekt med høy kontrast til bakgrunnen, gjør at normalisert magnitude til kantene vi leter etter blir lav.*

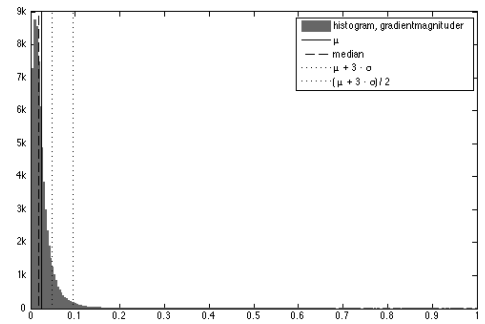
kanter man egentlig ønsker å detektere. Det har ikke blitt gjort eksperimentering med fjerning av outliers.

Undertrykke outliers En annen løsning kan være å undertrykke de høyeste verdiene i magnitudebildet ved å bestemme en øvre intensitetsverdi, der alle piksler med verdi over grensen settes ned til denne verdien. Dette gjør at pikslene beholder sin egenskap som høyeste magnituder i bildet, men at de nå deler denne egenskapen med endel flere piksler. Det har blitt prøvd å undertrykke outliers ved å sette grensen ved en persentil av gradientmagnitudens histogram. Eksempel på dette er vist i figur 7.19 (a)-(d), der det har blitt satt en øvre grense for gråtoneverdier ved 99.5-persentilen. Det har også blitt prøvd å gjøre en log-transform av bildet for å dempe de høyeste verdiene, men det er ikke tatt med figurer som illustrerer dette i rapporten.

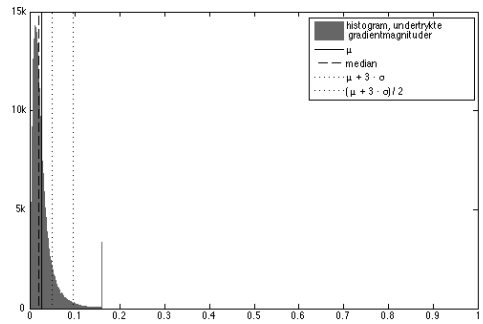
Statistiske egenskaper I stedet for å velge terskel ut fra høyeste verdi i magnitudebildet, er det et alternativ å bruke statistiske egenskaper som middelverdi, median og standardavvik. Terskel kan for eksempel settes ved middelverdi pluss et antall k standardavvik: $\mu + k\sigma$. Det kan være en fordel å undertrykke outliers i forkant av en slik operasjon, da outliers vil kunne påvirke både middelverdi og standardavvik. Median er mindre påvirkelig, og kan eventuelt brukes i stedet for middelverdi. Figur 7.19 viser histogram og gradientmagnitude for et undervannsbilde. I (a) ser vi at de aller fleste pikslene er punkter med lav magnitude. Det vil være ønskelig å sette grensen i området der kurven flater ut, da dette antas å være grensen mellom bakgrunn og forgrunn. Det har blitt forsøkt å terskle ved $\mu + 3\sigma$ og å gjøre en hysteresetersкling med øvre terskel lik $\mu + 3\sigma$ og nedre terskel som halvparten av dette, $\frac{\mu+3\sigma}{2}$. Resultatene av hysteresetersкling er vist i (e) og (f).

Estimere fordelingen til histogrammet Det har blitt forsøkt å tilpasse en fordeling til magnitudehistogrammet, og videre bruke fordelingens parametere til å definere terskler i stedet for gjennomsnittsverdi og empirisk varians. Ved å inspisere histogrammer ble det fremsatt en hypotese om at f_x og f_y, altså tettheten til magnituden i x- og y-retninger, var normalfordelte og at kvadratsummen av disse ville danne en Rayleigh-fordeling. Det ble da antatt at det på en nøyaktig måte kunne settes en terskel der fordelingen flater ut, enten ved å bruke middelverdi og standardavvik, eller ved å sette grensen ved en persentil. Rayleigh-fordelingens parameter σ ble bestemt ved å bruke uttrykket for middelverdi, $\mu = \sigma\sqrt{\frac{\pi}{2}}$, der gjennomsnittlig gradientmagnitude ble satt inn for μ. Parameteren σ går inn i uttrykket for varians, $\frac{4-\pi}{2}\sigma^2$, og dette ble brukt til å finne standardavviket.

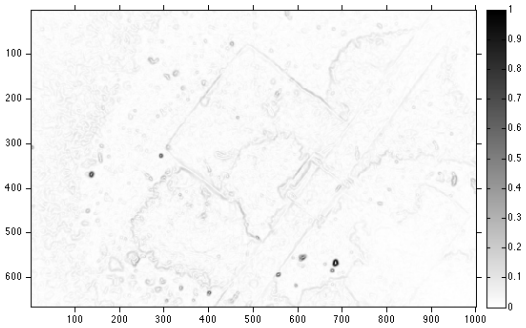
Figurene 7.20(a) og 7.20(b) viser histogrammene over gradientmagnitude i x- og y-retning for et undervannsbilde. De har en topp rundt 0, og flater deretter raskt ut. Det er imidlertid mulig å se at halene er endel tykkere enn hva som er tilfellet for en normalfordeling, så en ny hypotese er da at det er gradientene til *bakgrunnspikslene* som er normalfordelte, og at de tykke halene er et resultat av kanter i bildet. Derfor ble



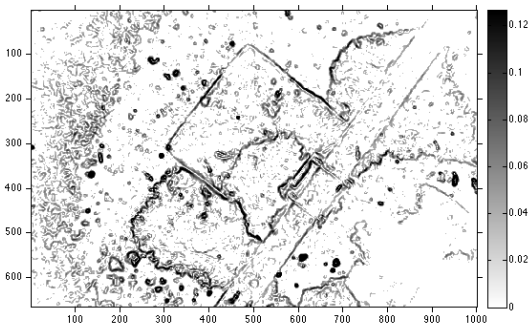
(a) *Histogram av gradientmagnituder*



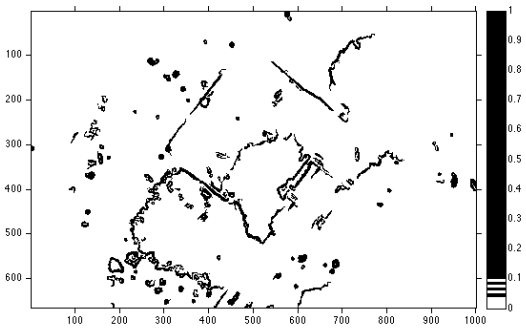
(b) *Outliers er undertrykt ved 99.5-persentilen*



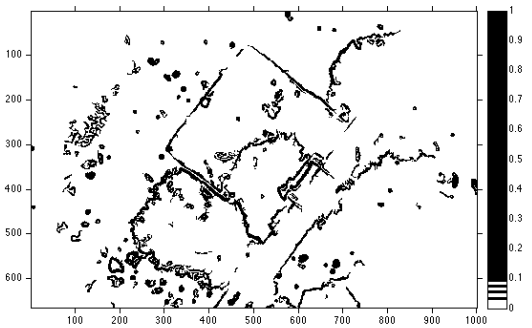
(c) *Gradientmagnitude*



(d) *Gradientmagnitude etter undertrykking av outliers*



(e) *Hystereseterskling med terskler fra statistiske egenskaper. Utregnede terskler fra* $\mu + 3\sigma$ *og* $\frac{\mu+3\sigma}{2}$ *er 0.115 og 0.057*



(f) *Utregnede terskler: 0.100 og 0.048*

Figur 7.19: *Histogram av gradientmagnituder før og etter undertrykking av outliers er vist i (a) og (b). Terskelverdier bestemt fra statistiske egenskaper er vist som prikkede linjer, satt ved* $\mu + 3\sigma$ *og* $(\mu + 3\sigma)/2$*. Gradientmagnituden er vist før og etter justering i (c) og (d). Tersklede resultater er vist i (e) og (f). For dette bildet ble de segmenterte kantene mer sammenhengende etter undertrykking av outliers.*

det forsøkt å klippe disse histogrammene ved 4 standardavvik, da dette skal inkludere 99.994% av verdiene i en normalfordeling. De vertikale linjene i figurene markerer grensene, og resultatene vises i figurene 7.20(c) og 7.20(d). Ved å ta kvadratsummen av disse to fordelingene ble resultatet som vist i figur 7.20(e).

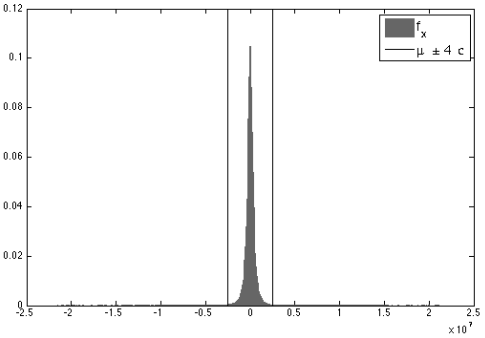
Eksperimentering viste at Rayleigh-fordelingen stemte for dårlig overens med fordelingen til gradientmagnituden til at det var hensiktsmessig å bruke denne for valg av terskelverdi. Fra dette ble det konkludert med at gradientkomponentene g_x og g_y ikke var normalfordelte, så å sette en terskel ut fra antagelsen om at gradientmagnitudene var Rayleigh-fordelt var ikke vellykket. Det finnes andre fordelinger med for eksempel tykkere haler som kunne vært bedre egnet til formålet, men det er ikke gjort noe videre arbeid med dette. Figur 7.20(f) viser eksempel på de to fordelingene lagt over hverandre.

Otsus metode

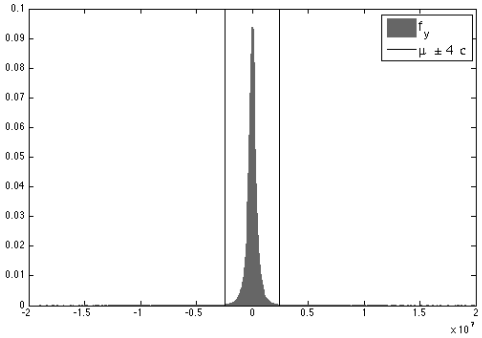
Som forklart i avsnitt 7.1.4 forsøker Otsus metode å finne en terskel ved å lete etter et punkt mellom to topper i et bimodalt histogram. På grunn av den store andelen bakgrunn (ikke kanter) i forhold til forgrunn (kanter) i magnitudebildene, og etter visuell inspeksjon av undervannsbildenes gradientmagnitudehistogram, er det grunn til å anta at også gradientmagnitudebildene er lite egnet for bruk av Otsus metode. Testing viste likevel at utregnet terskelverdi var brukbar i enkelte tilfeller, men at det totalt sett ikke var en pålitelig metode. Det er ikke lagt inn figurer som illustrerer denne testingen spesifikt.

Oppsummering

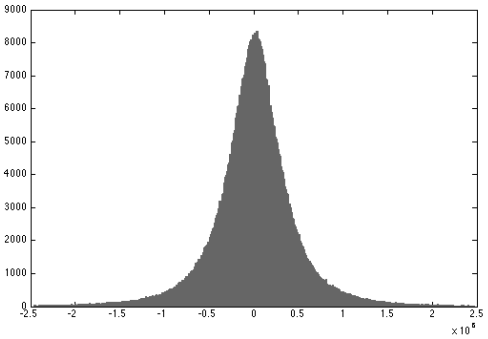
Av de ulike metodene som ble testet for å bestemme terskel for gradientmagnitude automatisk, har det vært vanskelig å finne noe som har fungert godt på alle treningsbilder. Å foreta en undertrykking av de aller høyeste verdiene har vist seg å være en nyttig operasjon for å gi de faktiske kantene i bildet en signifikant magnitude sammenlignet med øvrige verdier i histogrammet. Etter denne operasjonen har statistiske egenskaper fungert relativt greit for å finne terskelverdier, og kan være et alternativ å gå videre med. Likevel er det vanskelig å sette bestemte verdier for antall standardavvik som brukes i disse metodene. Når verdiene tilpasses til ett bilde, fører det gjerne i at resultatet blir dårligere i et annet.



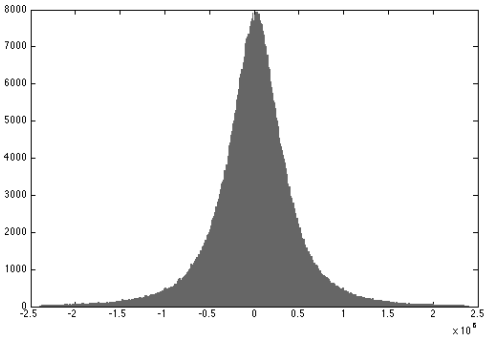
(a) *Histogram til* g_x



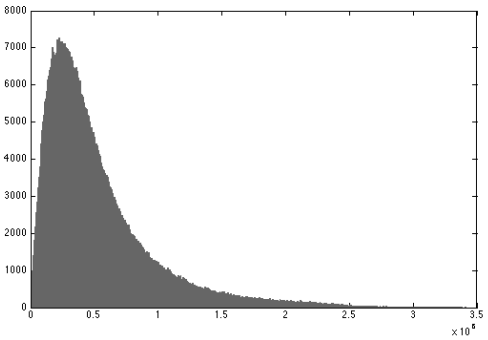
(b) *Histogram til* g_y



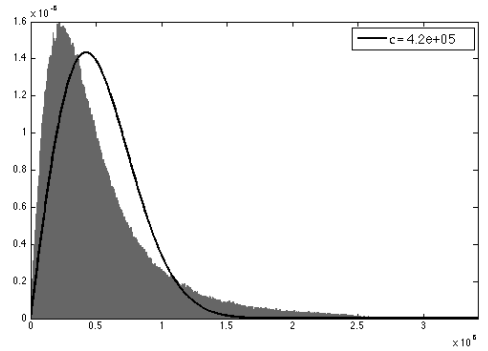
(c) *Histogram til* $g_x < \mu \pm 4\sigma$



(d) *Histogram til* $g_y < \mu \pm 4\sigma$

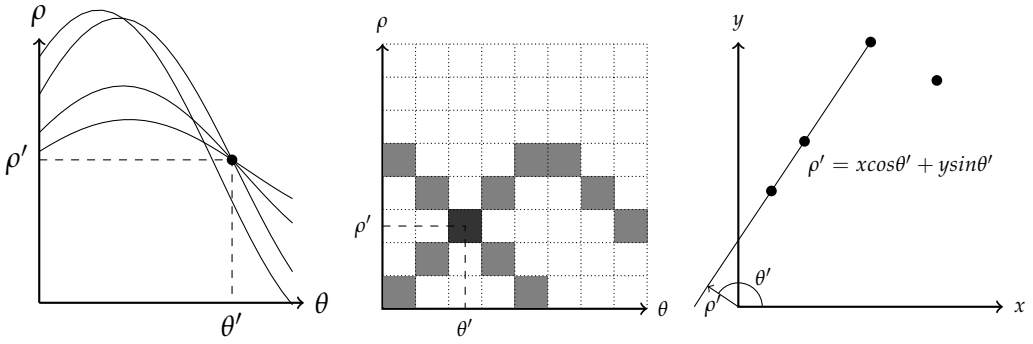


(e) *Histogram til kvadratsum av* $g_x < \mu \pm 4\sigma$ *og* $g_y < \mu \pm 4\sigma$



(f) *Fordelingen til gradientmagnitude sammenlignet med estimert Rayleigh-fordeling*

Figur 7.20: *Forsøk på å tilnærme histogrammet til bakgrunnsverdier til en Rayleigh-fordeling.*



(a) *Kurver og krysningspunkt i parameterrommet*

(b) *Akkumulatorceller i en houghmatrise*

(c) *Linjen plottet tilbake i bildeplanet*

Figur 7.21: *Houghtransform med ρθ-representasjon av linjene.*

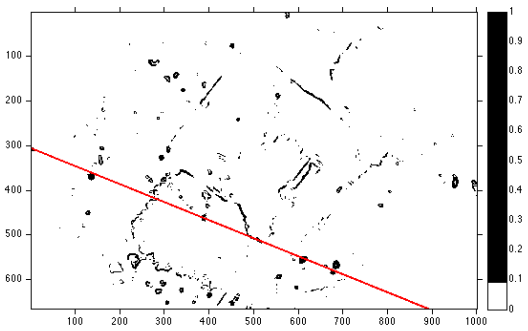
7.3 Houghtransform

I dette avsnittet omtales houghtransform [15] som er en praktisk metode for å detektere rette linjestykker i bilder. Det antas at teorien er kjent for leseren, men for å etablere en notasjon og enkelte sentrale begreper for resten av oppgaven inkluderes det i det følgende en kort beskrivelse av houghtransform med parametere ρ og θ. For en mer utdypende forklaring henvises det til litteraturen.

For hvert segmenterte piksel i et tersklet (binært) bilde kan det tegnes en kurve i det såkalte parameterrommet, eller $\rho\theta$-planet. Skjæringspunkter mellom kurvene i parameterrommet, lokalisert ved bestemte ρ og θ-verdier, definerer sannsynlige linjer i bildeplanet. Figur 7.21 illustrerer bruk av $\rho\theta$-representasjonen for linjer. For å detektere skjæringspunktene deles parameterrommet inn i diskrete celler (akkumulatorceller). Et utsnitt av en slik matrise er vist i figur 7.21(b). Når $\rho\theta$-kurvene plottes over matrisen, vil cellene under kurven øke sin verdi med 1 (illustrert med lys gråfarge i figuren). I celler hvor to linjer krysser hverandre vil verdien øke til 2, (mørkere gråfarge i figuren) og så videre. Skjæringspunktene detekteres ved å lete etter cellene med høyest verdi, og posisjonen angis med toppens ρ og θ-koordinater. En linje i bildet, definert av en topp i radonmatrisen, representeres ved hjelp av en vektor som starter i bildets origo og står normalt på linjen. Vektoren har lengde ρ og vinkel θ, der θ er gitt i antall grader fra x-aksen. For vertikale linjer blir da θ lik 0°, siden vektoren står normalt på linjen, og for horisontale linjer blir θ lik 90°. For hvert piksel i bildet plottes en kurve i $\rho\theta$-planet, og skjæringspunkter mellom kurvene definerer sannsynlige linjer.

7.3.1 Houghtransform av gradientbildet

I en houghtransform vil alle segmenterte piksler i et tersklet kantbilde telle like mye når verdiene i houghmatrisen akkumuleres, så i arbeidet med oppgaven har det blitt brukt mye tid på å finne gode måter å terskle gradientbildene på. Målet har vært å segmentere ut sammenhengende kanter som hører til eventuelle objekter, uten å få



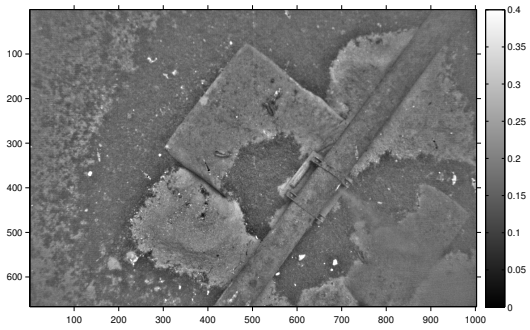
Figur 7.22: *I vanskelige bilder kan uønskede segmenterte piksler gi opphav til «falske» houghlinjer, her illustrert med en rød linje.*

med for mange kanter som stammer fra naturlige strukturer på havbunnen. Mange segmenterte piksler som tilfeldigvis ligger på en linje kan gi falske linjekandidater i houghmatrisen, noe som i verste fall kan overskygge bidraget fra de faktiske linjene i gråtonebildet. Figur 7.22 viser et eksempel på hvordan flere spredte punkter har gitt opphav til en «houghlinje». De fleste slike tilfeller er likevel ikke kritiske. Ved å anta at reelle kanter har en viss utstrekning, kan de minste segmenterte objektene fjernes før houghtransform ved hjelp av morfologiske operatorer eller `MATLAB`-funksjoner som `bwareaopen`. Man kan også gjøre en kvalitetssjekk av segmenterte objekter ved å for eksempel se på gjennomsnittlig gradientmagnitude for alle piksler i objektet. Dersom gjennomsnittet er for lavt kan objektet fjernes. I tillegg vil det uansett være behov for en verifikasjon av linjene som defineres av houghmatrisen. Slike houghlinjer har ingen definert start og slutt, så for å detektere linjesegmenter i bildet må alle bildepiksler langs houghlinjene inspiseres.

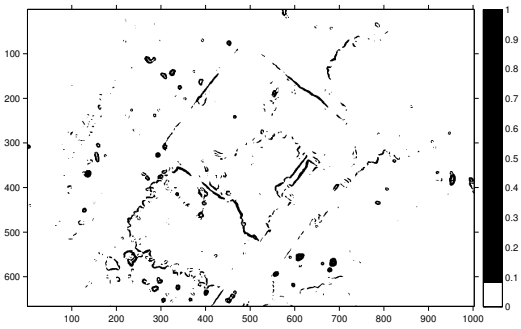
7.3.2 Radontransform av gradientbildet

På grunn av vanskelighetene med valg av terskel, ble det forsøkt å gjøre en ikke-binær houghtransform, det vil si at et piksels bidrag til houghmatrisen ble vektet med dets gradientmagnitude. Man kan anta at punkter som ligger langs tydelige kanter i bildet har høyere magnitude enn punkter fra øvrige områder i bildene, og dermed vil gi sterkest bidrag til houghmatrisen. En enkel implementasjon av dette har vært å benytte såkalt radontransform [10] i stedet for houghtransform. I praksis fungerer de to transformene på akkurat samme måte, ved å summere verdiene langs en gitt linje og lagre summen i en matrisecelle gitt av linjens parametere ρ og θ. Forskjellen er at houghtransformen krever at et piksels verdi er enten 0 eller 1 (typisk et tersklet bilde), mens radontransformen aksepterer alle verdier og dermed får en vektet sum langs linjen. Ulemper med denne fremgangsmåten er for det første at outliers kan få uforholdsmessig høy vekt, så det vil være fornuftig å undertrykke de høyeste verdiene slik som beskrevet i avsnitt 7.2.4. For det andre vil de aller fleste piksler nå bidra med verdier i radonmatrisen, ikke bare et fåtall av segmenterte

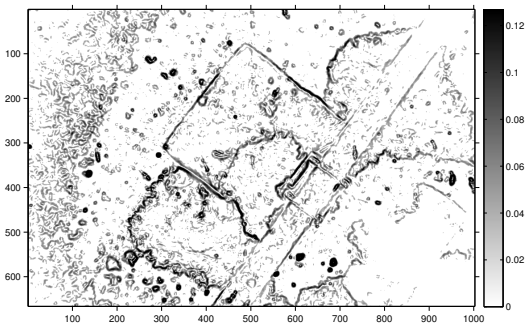
piksler som ved vanlig houghtransform. Selv om piksler med lav magnitude gir små bidrag, vil det store antallet føre til at flere celler i radonmatrisen får en viss verdi. Dette kan være med på å smøre ut toppene og gjøre det vanskeligere å definere eksakte lokale maksima. Derfor kan det også være aktuelt å sette en nedre grense for aktuelle magnitudeverdier. Figur 7.23 viser eksempler på hough- og radonmatriser. Resultatet etter houghtransform av et tersklet magnitudebilde er vist i (d) og direkte radontransform uten terskling er vist i (e).



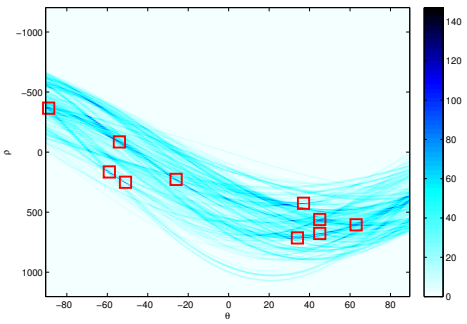
(a) *Homomorf-filtrert undervannsbilde. Gråtoneskala er komprimert fra [0, 1] til [0, 0.4] for bedre visning på papir.*



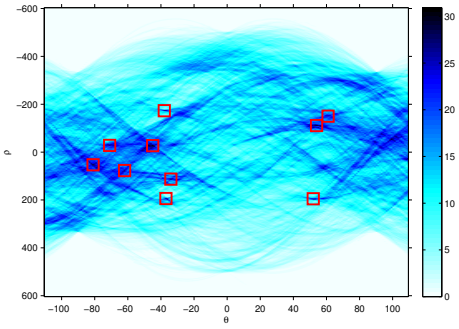
(b) *Sobelfiltrert og tersklet (0.1) versjon av (a).*



(c) *Sobelfiltrering av (a). Magnitudeverdier over 99-persentilen er undertrykt for bedre dynamikk.*



(d) *Matrise fra houghtransform av (b).*



(e) *Matrise fra radontransform av (c).*

Figur 7.23: *Hough- og radonmatriser. Lokale maksima er markert med røde rektangler. Siden houghtransform har definert origo i hjørnet av input-bildet, mens radon definerer origo i sentrum, blir det vanskelig å sammenligne matrisene visuelt.*

7.4 Kantdeteksjon ved fasekongruens

Det er visse ulemper knyttet til kantdeteksjon ved hjelp av gradientbaserte teknikker. De er blant annet følsomme for glatting, skalering og endringer i belysning. Glattingen benyttes for å fjerne støy før det utføres en kantdeteksjon, så det er åpenbart at dette påvirker metodene. Når det gjelder skalering kan det i praksis være er en utfordring å forutse bredden på kantene man ønsker å detektere og dermed hvilken størrelse man bør velge på filterkjernen. En og samme kant vil få ulike gradientmagnituder avhengig av filterstørrelse og skalering av bildet. Det kan også være problematisk å bestemme hvilke gradientverdier som svarer til kantene man leter etter og hvilke som er støy og bakgrunnsvariasjoner. Her kan belysning og kontrast spille en rolle. Det kreves en viss forskjell i gråtoneverdier for å detektere en kant, men hva som er riktig grense for ønskede og uønskede kanter kan være vanskelig å forutsi generelt for flere bilder.

En metode som kan brukes for å unngå disse problemene er fasekongruens [27], som betyr sammenfall eller overensstemmelse av fase. Metoden baserer seg på at på en linje eller på en kant vil Fourierkomponentene til signalet være i fase.

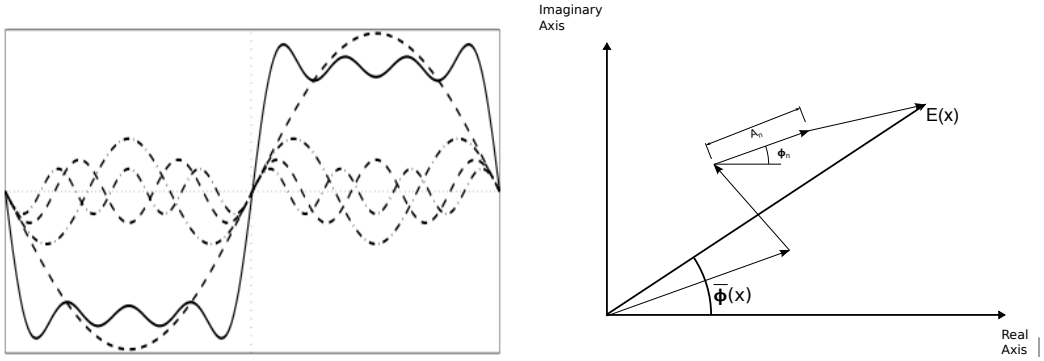
Figur 7.24(a) viser en firkantpuls, et signal som bytter raskt mellom to diskrete verdier. Ved å foreta en Fourier-rekkeutvikling av et slikt signal, kan det representeres som en sum av sinusfunksjoner. I figuren er dette illustrert som de stiplede linjene. Hver av disse sinussignalene kalles en Fourierkomponent, og legg merke til at i punktet hvor firkantpulsen har en flanke er alle Fourierkomponentene i fase. Det vil si at de har samme vinkel, samme fase: fasekongruens.

Et mål på fasekongruens er illustrert geometrisk i figur 7.24(b). De fire vektorene som er stilt etter hverandre er de ulike Fourierkomponentene $F(A, \Phi)$ til et signal hentet fra ett punkt, der A er amplitude og Φ er fase for en vektor. Lengden som er gitt av avstanden fra origo til enden av den ytterste vektoren kalles «lokal energi», $E(x) = \sqrt{\sum_n F_n^2}$. Fasekongruens PC kan uttrykkes som forholdet mellom lokal energi og summen av amplituden A_n til hver av komponentene. Dette er vist i likning (7.6).

$$PC(x) = \frac{E(x)}{\sum_n A_n(x)} \tag{7.6}$$

Fra denne definisjonen ser vi at dersom alle Fourierkomponenter er i fase vil uttrykket for fasekongruens maksimeres til 1. Hvis det ikke er noe samsvar mellom komponentene går verdien mot 0. Målet på fasekongruens påvirkes altså ikke av den totale magnituden til signalet. Dette gjør at det er lettere å finne kanter med lav kontrast, og det gjøres heller ingen forskjell på brede eller smale kanter.

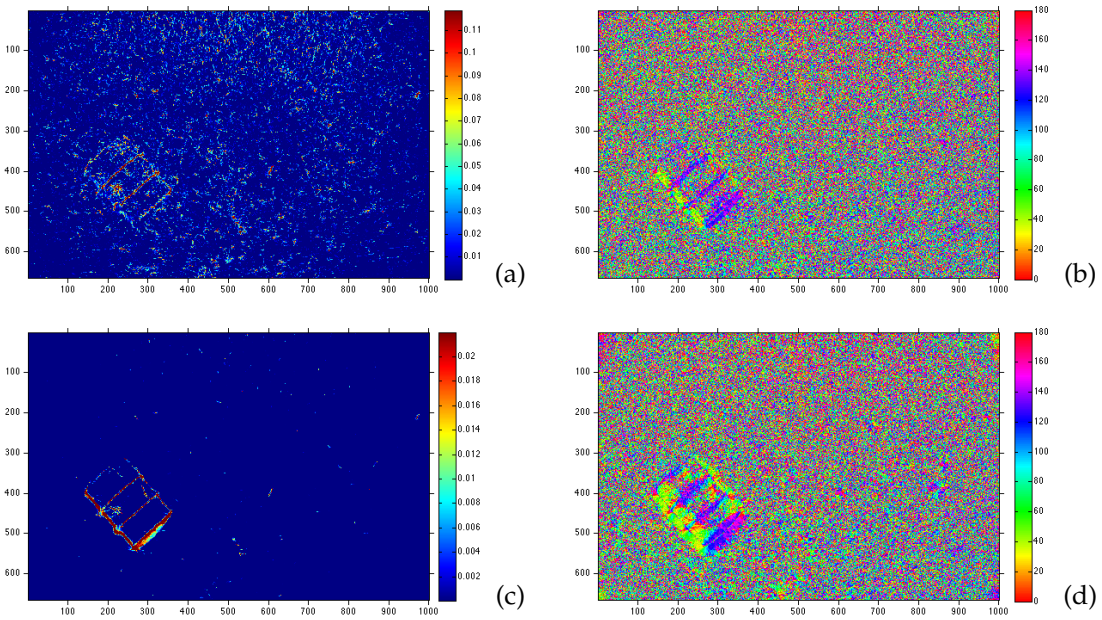
Eksempelet og definisjonen beskrevet over gjelder bare for endimensjonale signaler. For å kunne brukes i todimensjonale bilder sammenlignes filtrering fra flere forskjellige retninger, og resultatene brukes til å bestemme både magnitude og retning til bildepunktet. En utfyllende forklaring av algoritmen og implementasjonen er gitt i [27].



(a) *En signal som tilnærmer en firkantpuls (heltrukken linje) og fire av dens Fourierkomponenter (stiplede linjer) . Illustrasjon fra [29].*

(b) *Fourierkomponentene til et signal vist for ett punkt. Komponentene er plottet etter hverandre i et polarkoordinatsystem, og avstanden fra origo til enden av den ytterste vektoren gir lokal energi, $|E(x)|$. Illustrasjon fritt etter [29].*

Figur 7.24: *Fourierkomponentene til et signal vist over signalet selv (a) og komponentene satt sammen som ved utregning av fasekongruens (b).*



Figur 7.25: *Fasekongruens (venstre kolonne) og retning (høyre kolonne) for ulike parameterinstillinger. (a) og (b) viser resultatet ved standardinstillinger til filteret, mens (c) og (d) viser resultatet ved instillinger som brukt i oppgaven. Legg merke til at retning ser ut til å bli vilkårlig når fasekongruens er 0.*

Valg av parametere Implementasjonen av fasekongruens [28] som jeg har brukt i oppgaven gir mulighet til å stille på inntil ni ulike parametere for å styre filterets effekt. Parametrene er beskrevet i [27], artikkelen som implementasjonen bygger på. For å regne ut mål på frekvens og fase i lokale områder rundt punkter i bildet, brukes en wavelet-transform med logaritmiske Gabor-filtere. Parameterne gir mulighet til å velge antall filterorienteringer, antall filterskalaer og skaleringsfaktoren mellom dem, samt båndbredden til filtrene. Bølgelengden til filteret med laveste skala kan også defineres.

Parameteren k tilpasser filterets sensitivitet for støy. På grunn av normaliseringen som foretas i likning (7.6) ser vi at fasekongruensen kan bli høy, selv om magnituden til alle Fourierkomponentene er lave. Når signalet analyseres på en tilstrekkelig liten skala vil tilfeldige amplitudevariasjoner på grunn av støy kunne fremstå som relativt tydelige kanter, og målet på fasekongruens blir høyt. I algoritmen har det derfor blitt valgt å estimere støy nettopp ut fra det minste filteret. Det antas at bakgrunnsstøyen har et konstant effektspektrum og derfor vil filterresponsen være lik i alle områder av bildet, untatt der det finnes faktiske kanter og linjer. Gitt den lave utstrekningen til det minste filteret tas det utgangspunkt i at fordelingen til amplituderesponsen, sett over hele bildet, hovedsaklig er fordelingen til bakgrunnsstøy. Støy-energien regnes ut som en kvadratsum av to antatt uavhengige, normalfordelte variable og kan dermed modelleres som en Rayleigh-fordeling. Gjennomsnitt μ_R og standardavvik σ_R estimeres for fordelingen. Her kommer parameteren k inn, som bestemmer en terskel T gitt av $T = \mu_R + k\sigma_R$. Den gir altså mulighet til å bestemme hvor stor del av estimatet som faktisk skal regnes som støy. Støyen antas å være additiv, en vektor som kommer i tillegg til lokal energi (se figur 7.24(b)), så støy-energien subtraheres fra energivektoren $E(x)$ før normalisering. Et nytt mål på fasekongruens med støydemping blir som vist i likning (7.7), der $\lfloor\ \rfloor$ gir verdien 0 hvis uttrykket mellom klammene er negativt (støy er kraftigere enn signal). For å håndtere tilfeller hvor alle Fourier-amplituder er lave og verdien under brøkstreken blir nær 0, legges det til en liten positiv konstant ϵ.

$$PC(x) = \frac{\lfloor E(x) - T \rfloor}{\sum_n A_n(x) + \epsilon} \tag{7.7}$$

Fasekongruens vil nå være et mål på styrken til en egenskap relativt til støynivået omkring.

I punkter med få frekvenskomponenter med amplitude av betydning blir målet på fasekongruens lite informativt. I et særtilfelle med bare én frekvenskomponent vil, ut fra uttrykket, fasekongruensen alltid være 1. Selv om det finnes flere frekvenskomponenter vil verdien av uttrykket også gå mot 1 hvis komponentene er tilnærmet lik hverandre. En glatting av et signal vil redusere spennet av frekvenser og ved ekstrem glatting vil signalet til slutt kunne tilnærmes med en enkelt frekvenskomponent. I algoritmen er det derfor innført et normalisert mål på spredning av frekvenser, $s(x)$, som går fra 0 til 1. Lav frekvensspredning straffes ved å vekte fasekongruensen med en sigmoidfunksjon

$$W(x) = \frac{1}{1 + e^{\gamma(c - s(x))}} \tag{7.8}$$

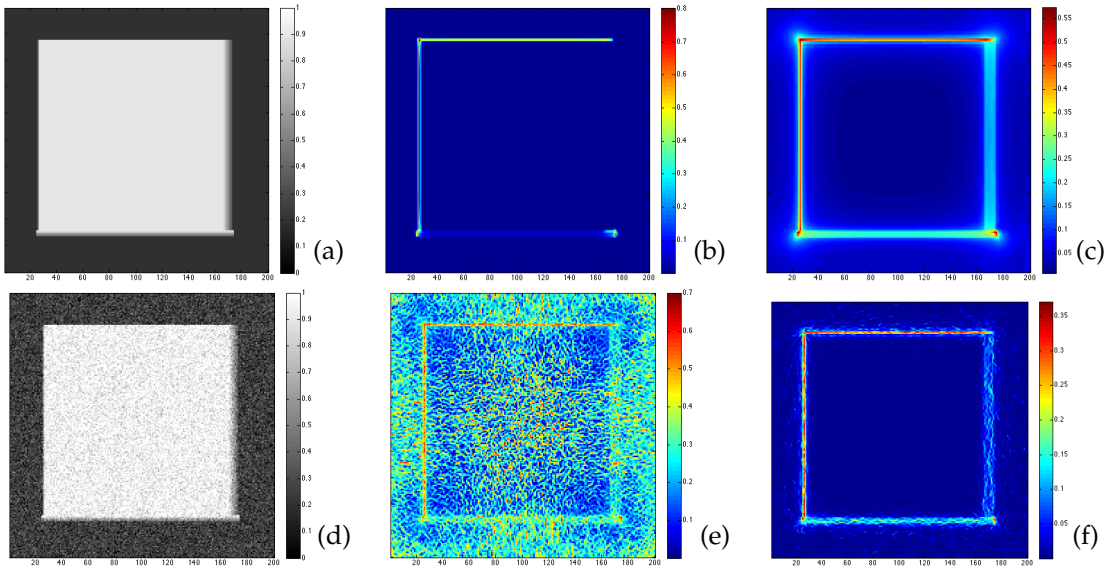
der c er knekkfrekvens til sigmoidfunksjonen og γ bestemmer hvor brått funksjonen knekker. Parameteren c setter et minstemål for hva som regnes som stort nok spenn av frekvenser, mens γ definerer hvor skarpt dette skillet blir. I MATLAB-funksjonen kalles disse parametrene for `cutOff` og g.

Uttrykket for fasekongruens utvides ytterligere i artikkelen, både for økt robusthet/nøyaktighet og for å kunne tas i bruk på todimensjonale signaler, som bilder. Jeg har valgt å begrense beskrivelsen til det som er inkludert hittil, da dette gir et tilstrekkelig innblikk i hva de ulike parametrene virker inn på.

Ulike verdier har blitt testet for de fleste av parametrene, og et syntetisk bilde har blitt brukt for å teste virkningen av de ulike innstillingene. Figur 7.26(a) viser et kvadrat med gråtone 0.9 på en bakgrunn med gråtone 0.2. Kantene til rektangelet har fått ulike bredder ved å variere gråtoneverdien lineært i overgangene mellom rektangel og bakgrunn: På øvre, venstre, nedre og høyre side er kanten henholdsvis 1, 3, 6 og 9 piksler bred. I figur 7.26(b) vises resultatet av en fasekongruensfiltrering med standard parameterinnstillinger, og vi ser at høyre kant i dette tilfellet har blitt fullstendig ignorert. Hvis både c og γ fra likning (7.8) senkes, vil brede, lavfrekvente kanter aksepteres i større grad. Et eksempel på dette er vist i figur 7.26(c), der c (`'cutoff'`) er justert fra 0.5 til 0.2 og γ (`'g'`) er justert fra 10 til 5. Et annet alternativ er å øke `'nscales'` for å lete etter brede kanter. I figur 7.26(d) har det blitt lagt til støy med `imnoise(im,'gaussian',0.05)`. Dersom all støyfjerning utelates (parameter `'noisemethod'= 0`) viser figur 7.26(e) at det detekteres kanter over alt. Støyfjerning med `'noisemethod'= -1` og `'k'= 4` forbedrer resultatet betraktelig, som vist i figur 7.26(f). Også ved å justere parameteren `'minwavelength'` kan man styre bredden til de tynneste detekterte kantene, da denne minste bølgelengden brukes i utregningen av støy.

En opplisting av mine parametervalg er vist nedenfor. (Merk at parameternavnene kan forkortes). Jeg har valgt å øke `'nscales'` for å detektere bredere kanter og kanter med lav kontrast. `'minwavelength'` er redusert sammen med en økning av `'k'` for å bare fjerne de aller tynneste kantene. `'mult'` er tilpasset `'sigmaonf'` for å få en jevn dekning av hele frekvensspekteret (ref. dokumentasjonen til filteret). `'cutoff'` og `'g'` er redusert for å inkludere kanter med lav kontrast og bredere kanter.

```
pc_param = [...
  {'nscales' } {6} ...   %(4)   Number of wavelet scales
  {'norient' } {6} ...   %(6)   Number of filter orientations
  {'minwave' } {3} ...   %(3)   Wavelength of smallest scale filter (px)
  {'mult'    } {3} ...   %(2.1) Scaling factor between successive filters
  {'sigmaonf'} {.55} ... %(.55) Affects bandwidth of filters
  {'k'       } {5.0} ... %(2)   Noise thresh. # of stddev of noise energy
  {'cutoff'  } {0.25}... %(0.5) Fractional measure of freq spread. penalty
  {'g'       } {5} ...   %(10)  Sharpness of transition in sigmoid function
  {'noisem'  } {-1} ...  %(-1)  Noise method or fixed threshold
];
pc = phasecong3(I, pc_param{:});
```



Figur 7.26: *Eksperimentering med parametervalg for fasekongruens.*
(a): Konstruert rektangel med kanter av ulike bredder, (b): fasekongruensfiltrering med standard parametere, (c): bredere kanter er forsøkt inkludert, (d): hvit støy er lagt over bildet, (e): all støyfjerning er utelatt i parameterinnstillingene, (f): hvit støy er forsøkt fjernet.

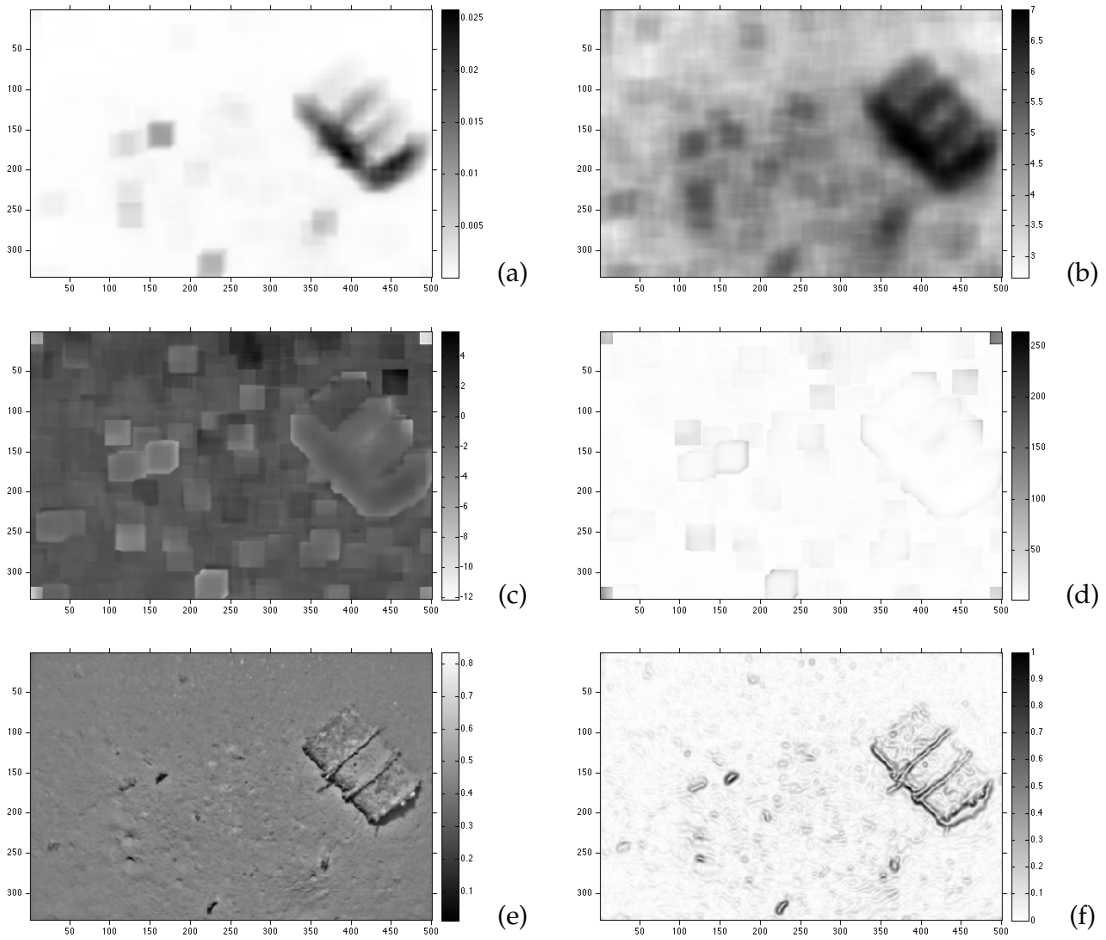
7.5 Teksturer

I kapittel 3 ble det nevnt at tekstur var et mulig alternativ for å detektere objekter. Dette kan være aktuelt når man antar at objektenes overflatetekstur vil skille seg fra havbunnen, og hvis dette er tilfelle bør man relativt lett kunne segmentere ut objektet i sin helhet. Dersom det i oppgaven velges å bruke kantdeteksjon (avsnitt 7.2) for å beskrive egenskapene til objekter, kan en foregående segmentering fra tekstur gi et begrenset søkeområde eller interesseområde for gradientoperatoren.

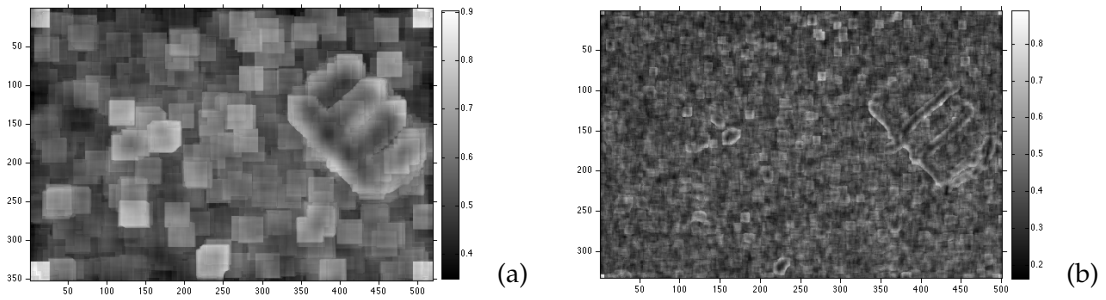
Som man kan se av datasettet presentert i kapittel 4, er havbunnen og eventuelle objekter i flere tilfeller dekket av et mudderlag som gjør at objektene går relativt i ett med omgivelsene. Det er i hovedsak kantene som definerer gjenstandene, og en antagelse er at det i beste fall er disse som kommer frem ved en teksturanalyse. For bakgrunnsstoff om deteksjon av teksturer og om andre regiondeskriptorer henvises det til [4, 15].

Tekstur kan detekteres ved å se på førsteordens statistikk fra gråtonehistogrammer lokalt i bildet. På et subsett av bilder ble det testet regionglatthet (et normalisert mål på varians), entropi, histogramskjevhet og histogramkurtose. Som forventet viste egenskapsbildet etter glatthet-filtrering høyest verdier omkring skyggekantene til objektene. Skjevhet og kurtose var generelt sett lite informative egenskaper, mens entropi ga i flere tilfeller relativt gode indikasjoner på objektenes plassering. Figur 7.27 viser resultater for ett bilde. Det er brukt vindusstørrelse 31 x 31 for alle teksturfiltere. For utregning av egenskapene ble det brukt MATLAB-funksjonene `entropyfilt` og `colfilt` med `@var`, `@skewness` og `@kurtosis`. Til sammenligning er originalbildet og gradientmagnitude fra sobelfiltrering vist i figuren. Dette viser at teksturfiltreringene har gitt lite ekstra informasjon i forhold til kantdeteksjonsalgoritmer.

En vanlig måte å detektere tekstur på, er med andreordens statistikk fra forhold mellom pikselpar. Denne informasjonen hentes ut gjennom såkalte naboskapsmatriser, eller *Gray-Level Co-occurence Matrices (GLCM)* [15]. Bruk av glidende vindu og GLCM for egenskapsuttrekking er en tidkrevende prosess. På min datamaskin tok det omkring 3 minutter og 50 sekunder å regne ut homogenitet med vindusstørrelser 9x9 og 31x31 på bilder med dimensjon 333x501. Resultatet av disse filtreringene er vist i figur 7.28. Kildebildet er det samme som vist i figur 7.27(e). Homogenitet ble antatt å være en nyttig egenskap for å skille glatte objektoverflater fra naturlig havbunn, men som figuren viser er oljefatet dekket av det samme mudderet som havbunnen, så i dette tilfellet forventes det ikke at objektet kan detekteres ved å skille på tekstur. Den lange prosesseringstiden er også uheldig når målet er å analysere store datasett, så bruk av tekstur i forbindelse med deteksjon av objekter i undervannsbilder antas å være lite anvendelig av flere årsaker.



Figur 7.27: *Deteksjon av teksturer ved hjelp av de førsteordens egenskapene glatthet (a), entropi (b), histogramskjevhet (c) og histogramkurtose (d) utregnet i glidende vinduer av størrelse 31 x 31 piksler. Originalbilde (e) og gradientmagnitude (f) er vist for sammenligning.*



Figur 7.28: *Grad av homogenitet målt i bildet ved hjelp naboskapsmatriser (GLCM) i glidende vindu. Vindusstørrelser er 31x31 (a) og 9x9 (b).*

7.6 Valg av metode

I de foregående seksjoner har jeg sett på ulike metoder for å detektere eller lokalisere mulige objekter. I dette avsnittet argumenteres det for valget av hvilke metoder som brukes videre i oppgaven.

Flere av de skyggebaserte metodene førte til god segmentering av skygger og kanter uten at for mye støy og for mange uønskede elementer ble inkludert, men av flere årsaker valgte jeg likevel å ikke gå videre med skyggedeteksjon. For det første var resultatene veldig avhengig av gode terskelvalg, og det var vanskelig å gjøre disse valgene automatisk på en robust måte. De adaptive metodene var også følsomme for valg av parametere og var i tillegg relativt tidkrevende. For det andre vil det være en fordel å kunne lete etter mer enn bare skygger eller mørke kanter, da objekter hvor kantene har lav kontrast i forhold til havbunnen vil være vanskeligere å avdekke. Gjennom arbeidet med skyggedeteksjon fikk jeg eksperimentert en del med ulike former for støyfiltrering, bakgrunnsutjevning og terskling som jeg også brukte i eksperimenteringen med de gradientbaserte teknikkene.

Kantdeteksjon med Sobel gradientoperator fungerte bra, og ved å øke størrelsen på filterkjernen forsvant mange uinteressante kanter som stammet fra ujevnheter på havbunnen. Den store filterkjernen førte i midlertid til at detekterte kanter ble bredere. Generelt for gradientbaserte metoder gjelder det også at tynne linjer i et bilde vil gi doble kanter i gradientbildet, noe jeg vurderte som uheldig med tanke på videre prosessering med houghtransform. Dette omtales mer i detalj i avsnitt 7.6.1. Cannys kantdetektor er en mer avansert algoritme som i etterkant av glatting og kantdeteksjon med gradientoperator gjennomfører en tynning av kantene ved å markere piksler der det finnes lokale maksima i gradientmagnitude. Disse punktene lenkes deretter sammen ved hjelp av hysteresetterskling. Dette gir både tynne og sammenhengende kanter, men fortsatt doble kanter for tynne linjer. Det er kjent at Laplace-operatoren kan brukes for å unngå de doble kantene, men det har ikke blitt gjort eksperimenter med det i oppgaven. Som med de skyggebaserte metodene var det vanskelig å gjøre en automatisk terskling av gradientmagnitude.

Fasekongruens var en metode som egnet seg godt for kantdeteksjon i undervannsbilder. Siden undervannsbilder ofte har lav kontrast og fordi sedimenter kan legge seg over objektene og gjøre kantene utydelige, vil de gradientbaserte teknikkene kunne ha problemer med å detektere alle kanter. I disse tilfellene har fasekongruens vist seg å være mer robust. En annen fordel er at tak-kanter ikke gir dobbel respons, slik som ved bruk av Sobel og Canny. En ulempe med fasekongruens er at filteret er relativt langsomt (i `MATLAB`) og at det er mange parametrene som kan påvirke filterets effekt. En systematisk testing og en grundig studie av dokumentasjonen gjorde det likevel mulig å finne et sett av parametere som fungerte godt på de fleste bildene i treninggsettet. Jeg har valgt å bruke fasekongruens som kantdetektor.

Kantbevarende glatting ved hjelp av anisotrop diffusjon var en interessant filtreringsmetode, som ble antatt å være godt egnet til problemstillingen. Filteret kan utføre kraftig glatting uten å viske ut kanter omkring objekter i bilder, men dette gjelder selv-

følgelig også for ikke-menneskeskapte objekter med høy kontrast til havbunnen, som for eksempel marin snø. Filteret er relativt sensitivt i forhold til kontrastparameteren κ, som setter en grense for hvilke gråtonedifferanser som regnes som kanter i bildet. Siden kantene jeg ønsker å finne kan ha en forholdsvis lav kontrast, og fordi kantdeteksjon med både sobel gradientoperator, Cannys algoritme og fasekongruens alle har en innebygd støyfiltrering, valgte jeg å ikke gå videre med bruk av filteret.

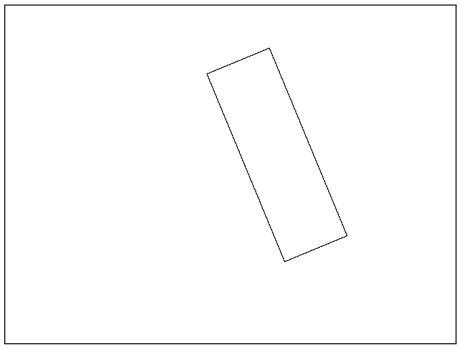
Jeg valgte å bruke homomorf filtrering som bakgrunnsutjevning først og fremst for å øke kontrasten nær bildekantene, men også fordi ujevn belysning påvirker de faktiske kantene i bildet, som nevnt i avsnitt 7.2.1. Jeg kuttet ut bottom-hat fordi transformen førte til ekstra strukturer på havbunnen (falske kanter) og fordi det ikke var nødvendig med både homomorf filtrering og bottom-hat transform.

Teksturer ble først og fremst valgt bort på grunn av dårlige resultater sammenlignet med de andre metodene. Den lange prosesseringstiden ble også vurdert som en ulempe.

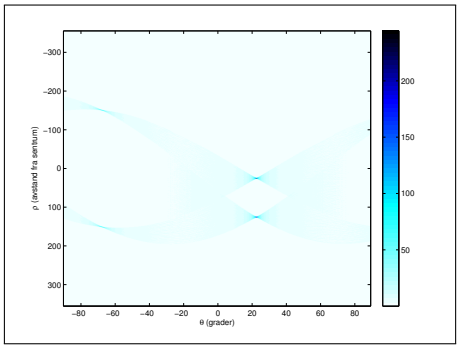
For å detektere rette linjer i bildene har jeg prøvd å benytte houghtransform. Input til houghtransform har vært tersklede gradientmagnitudebilder, dannet ved hjelp av Sobelfiltrering eller Cannys algoritme. Jeg har brukt endel tid på å finne gode metoder for terskling av gradientmagnituden, slik at jeg får et tersklet magnitudebilde med sammenhengende kanter og lite støy. Best resultat har blitt oppnådd ved bruk av hysteresetterskling, men robuste valg av terskelverdier viste seg å være utfordrende. Å unngå terskling av kantbildene og heller gjøre en radontransform direkte viste seg å være svært vellykket. Ved testing ga ekte kanter i bildene alltid opphav til de høyeste toppene i radonmatrisen.

7.6.1 Radontransform av gradientmagnitude og fasekongruens

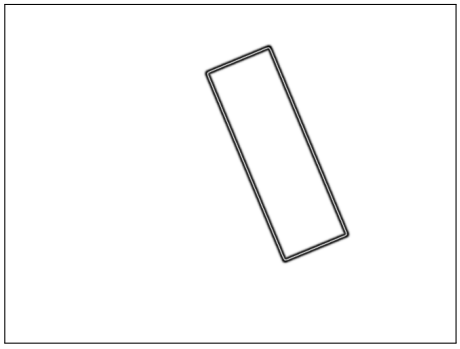
Som nevnt i forrige avsnitt, er en ulempe med gradientbaserte metoder den doble responsen for tynne kanter. I figurene 7.29 og 7.30 har jeg laget et rektangel og sammenlignet radontransform etter sobelfiltrering og fasekongruens. Vi ser at det sobelfiltrerte bildet har to kanter omkring rektanglet, som gir opphav til to topper i radonmatrisen i figur 7.30(a). På grunn av masken som legges omkring hvert lokale maksimum i radonmatrisen blir bare den ene toppen detektert, selv om begge er like riktige. I figur 7.30(c) ser vi hvordan deteksjonen blir unøyaktig, siden linjen man ønsker å detektere egentlig ligger mellom de to kantene i magnitudebildet. Den røde pilen er vektoren definert av toppen i radonmatrisen og den røde streken er linjen definert av vektoren. For fasekongruens blir det derimot bare en topp i radonmatrisen, og den detekterte linjen samsvarer bedre med innholdet i bildet, som vist i (d) og (f). Dette talte for å benytte fasekongruens til kantdeteksjon. For å unngå doble kanter kunne jeg også brukt Laplace-operatoren, men denne ble valgt bort til fordel for fasekongruens, fordi jeg ønsket å unngå gradientbaserte teknikker.



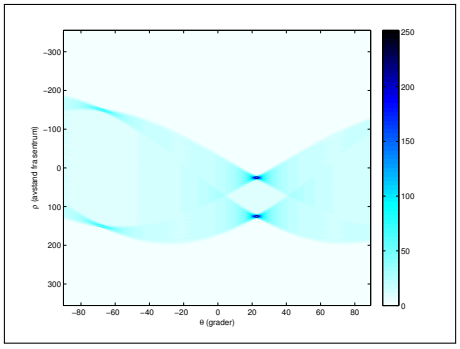
(a) *Rektangel rotert 22.5 grader*



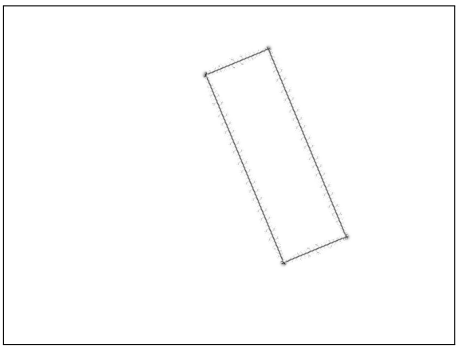
(b) *radontransform av rotert rektangel*



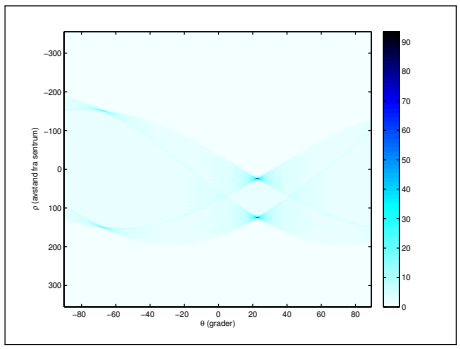
(c) *Sobelfiltrert, filterkjerne 15x15*



(d) *radontransform av sobelfiltrert*

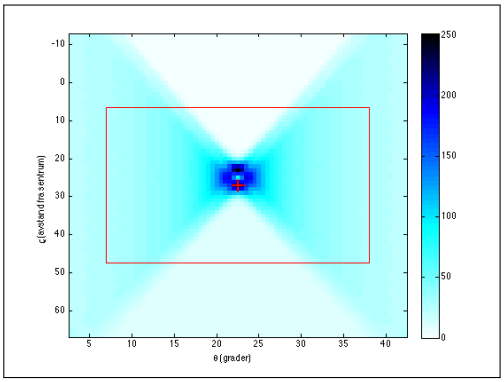


(e) *Fasekongruens: standard parameterinnstillinger.*

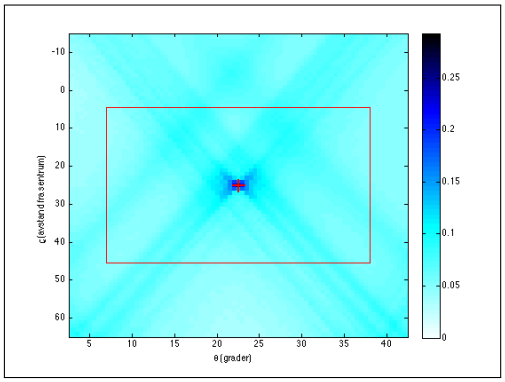


(f) *radonmatrise av fasekongruens*

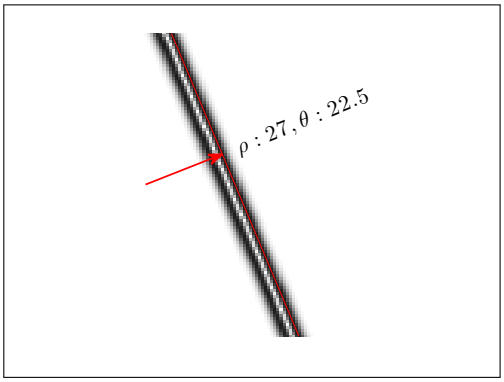
Figur 7.29: *Sammenligning av sobelfiltrering og fasekongruens*



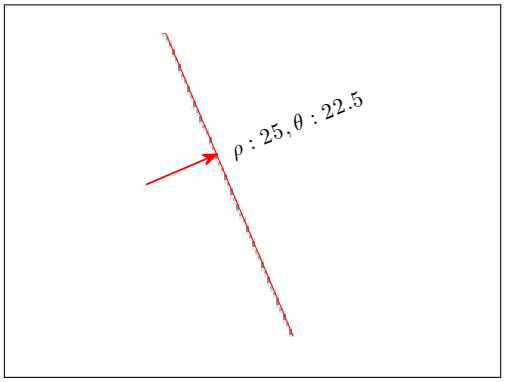
(a) *Utsnitt av radonmatrise for sobelfiltrert bilde. To topper, men masken undertrykker den ene.*



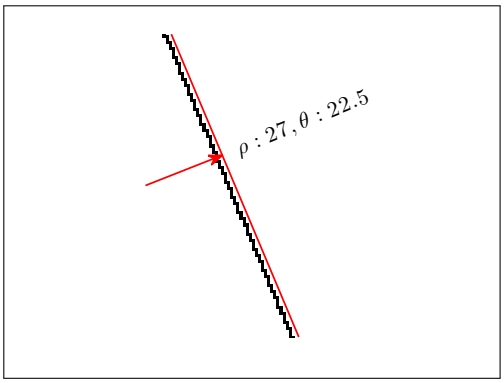
(b) *Utsnitt av radontransform for fasekongruensfiltrert bilde. En enkelt topp.*



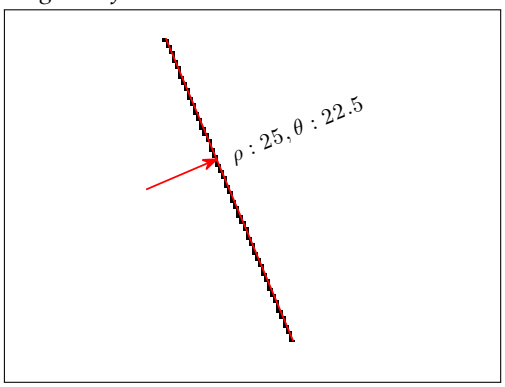
(c) *Detektert linje fra (a) plottet over sobelfiltrert bilde (utsnitt).*



(d) *Detektert linje fra (b) plottet over fasekongruensfiltrert bilde (utsnitt).*



(e) *Detektert linje fra (a) plottet over originalbilde (utsnitt).*



(f) *Detektert linje fra (b) plottet over originalbilde (utsnitt).*

Figur 7.30: *Ulempe ved bruk av sobelfilter i tilfeller med tynne linjer*

Kapittel 8

Utvikling av detektoren

De foregående kapitler har i hovedsak handlet om valg av metoder for preprosessering og forsøk på å gjøre en robust skyggesegmentering eller kantdeteksjon i undervannsbilder. I dette kapittelet beskrives videre prosessering og utviklingen av en detektor for menneskeskapte objekter i undervannsbilder.

Jeg har valgt en metode som ser på detekterte kanter fra bildene og videre sammenligner to og to kanter for å se om disse danner hjørner i form av rette vinkler, eller om de er parallelle og står overfor hverandre slik at de kan tenkes å danne et rektangel. Deteksjonsalgoritmen kan deles inn i følgende trinn:

1. Preprosessere og utføre geometrisk korreksjon
2. Utføre kantdeteksjon ved hjelp av fasekongruens
3. Detektere generelle linjer i bildene med radontransform
4. Ekstrahere linjesegmenter som svarer til faktiske kanter eller linjer i bildene
5. Finne par av linjesegmenter som kan danne hjørner eller paralleller
6. Gruppere flere par av linjesegmenter til større objektdeteksjoner

Avsnitt 8.1 oppsummerer i korte trekk hva som er utført av preprosessering og analyse på bildene for å kunne bruke dem som input til detektoren. Dette omfatter alt frem til og med radontransform. Avsnitt 8.2 beskriver overgangen fra topper i radonmatrisen til generelle linjer i bildet. I avsnitt 8.3 analyseres disse linjene for å trekke ut linjesegmenter, basert på egenskaper i kantbildene. Linjesegmentene skal svare til faktiske kanter i bildene. Avsnitt 8.4 handler om hvordan detekterte linjesegmenter/kanter settes sammen til større objektdeteksjoner og et mål på tiltro til hver deteksjon omtales i avsnitt 8.5. I avsnitt 8.6 vises et flytskjema med hovedfunksjonene i programkoden og en tabell med brukerstyrte parametere som vil kunne påvirke ytelsen til detektoren. I avsnitt 8.7 introduseres ROC-kurver som verktøy for evaluering av detektoren, og valg for utarbeidelse av fasit diskuteres. I avsnitt 8.8 utføres det til slutt eksperimenter for å evaluere noen mulige valg av parametere til detektoren.

8.1 Oppsummering av preprosessering

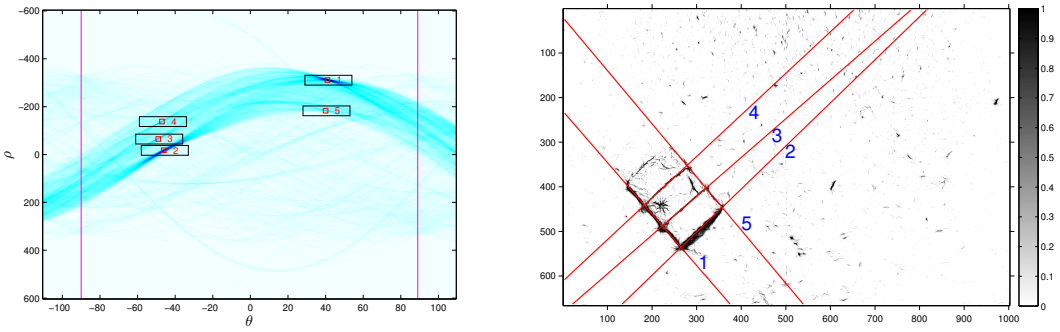
Målet med preprosesseringen er å håndtere noen av de kjente problemer ved undervannsavbildning, slik at bildene blir egnet for videre analyse. Etter innlesing av bildet brukes homomorf filtrering for å jevne ut bakgrunnsbelysningen. Filtreringen øker også kontrasten i bildet, som er en fordel med tanke på senere kantdeteksjon. Videre utføres en geometrisk korreksjon for å kompensere for linseforvrengning, slik at linjer som er rette i virkeligheten blir rette også i bildet. Til linjedeteksjon brukes fasekongruens, som er robust både i forhold til dårlig kontrast, svak kantstyrke og ulike kantbredder. Filteret gir ut et egenskapsbilde som angir grad av fasekongruens, en indikasjon på tilstedeværelse av kanter og linjer i bildet. Filteret kan også produsere et egenskapsbilde som indikerer hvilken retning de detekterte kantene har. Fasekongruensbildet brukes som input til radontransform for å detektere dominante linjer i bildet. De øvrige avsnitt i dette kapittelet omtaler den videre prosesseringen fra radonmatrisen.

8.2 Representasjon av radonlinjer

Som beskrevet i avsnitt 7.3, svarer toppene i radonmatrisen til overordnede, rette linjer i bildet som knytter sammen piksler med høy fasekongruens. Matrisekoordinatene ρ og θ definerer bare en normalvektor til disse teoretiske linjene, så det kreves videre regning for å finne ut hvor linjene passerer bilderammen og hvilke pikselkoordinater som ligger under linjene. For en linje er dette løst ved å finne pikselkoordinatet (x, y) på linjen, gitt av $[\rho, \theta]$-vektoren. I linjens retning, $\theta + 90°$, plukkes det ut to nye punkter $(x_a, y_a), (x_b, y_b)$ i maksimal avstand fra koordinatet (x, y). Med origo midt i bildet vil denne maksimale avstanden være lik halvparten av bildets diagonal. De tre punktene ligger alle langs den teoretiske linjen, med (x, y) midt mellom (x_a, y_a) og (x_b, y_b), der de to siste vil ligge på eller utenfor bilderammen. Alle pikselkoordinater mellom (x_a, y_a) og (x_b, y_b) hentes ut ved hjelp av Bresenhams algoritme [7, 42], og av de utregnede pikselkoordinater tas det vare på de som er gyldige innenfor bildets dimensjoner. Fra denne operasjonen får man både hvor linjene passerer bilderammen og hvilke pikselkoordinater som ligger under de teoretiske linjene. Figur 8.1 viser radonmatrisen og de utvalgte radonlinjene for et fasekongruensbilde. I enkelte bilder har det oppstått noen artifakter fra selve avbildningen. Dette er vertikale linjer enten midt i eller helt ute ved kantene til bildet. Det er lagt inn en sjekk for dette i toppdeteksjonen, slik at disse linjene kan ignoreres.

8.2.1 Toppdeteksjon i radonmatrisen

Det finnes flere alternativer for å detektere lokale maksima i radonmatrisen. `MATLAB` sin funksjon `houghpeaks` leter først etter høyeste verdi i matrisen. Punktet registreres og så settes verdien i punktet og i et område omkring til verdien 0. Dettes gjentas iterativt til ønsket antall topper er detektert.



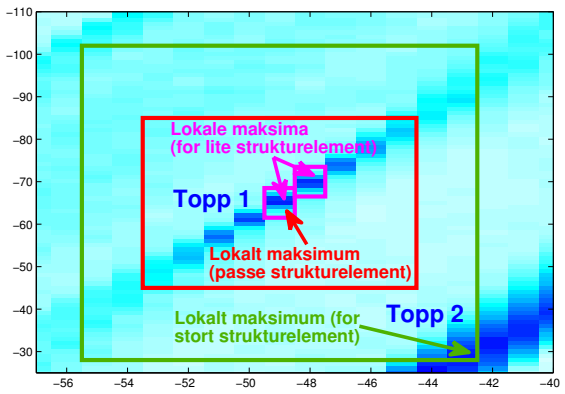
(a) *Radonmatrise* (b) *Radonlinjer over fasekongruensbilde*

Figur 8.1: *Radonmatrise og radonlinjer i et fasekongruensbilde. Toppene i (a) er nummerert i synkende rekkefølge ut fra høyde, og svarer til linjene i (b). De vertikale linjene i (a) marker grensene for gyldige θ-verdier, -90° til 89°. De største rektanglene indikerer områder omkring hver topp som undertrykker andre deteksjoner.*

Jeg har valgt å bruke en morfologisk metode til toppdeteksjon, hvor det utføres en gråtonedilatasjon av matrisen `I` med et strukturelement hvor senterpikselet er 0. Lokale maksima kan da detekteres ved å se hvor den opprinnelige matrisen har høyere verdi enn den dilaterte. Når strukturelementet ligger over pikselet med aller høyeste verdi vil altså dette pikselet tilordnes verdien til naboen med høyest verdi. Pikselet selv tas ikke med i beregningen, og `I > dilatert(I)` vil være sant for dette pikselet. Nabopikslene vil derimot få verdien til nettopp dette maksimalpunktet, og `I > dilatert(I)` vil være usant for disse. På grunn av matrisens diskrete representasjon dannes det lokale maksima på mange steder i bildet, og det må velges et relativt stort strukturelement for å ikke «lures» av dette. Topper i matrisen defineres av punkter hvor flere sinuskurver overlapper hverandre, og det er naturlig at slike topper får noe utstrekning.

Det ble gjort et litteratursøk etter andre metoder for toppdeteksjon i radonmatrisen, uten å finne noe av særlig interesse. De fleste treff handlet om metoder som forsøker å angi toppene mer nøyaktig enn bare ut fra cellen med høyest verdi, såkalt subpiksel-nøyaktighet. Noe lignende kunne forøvrig vært nyttig også i denne oppgaven. Ikke nødvendigvis det å definere en topp med subpiksel-nøyaktighet, men heller det at en topp gjerne defineres mer av et område enn av et piksel, og at sann topp kanskje ligger litt til siden for pikselet med høyest verdi.

Uansett om man velger `MATLAB` sin metode eller metoden basert på morfologi, må det velges en størrelse på området som skal dempes rundt hver detekterte topp. En ulempe med den første metoden er at det kan dannes nye lokale maksima på grensene mot et område som har blitt satt til 0. Dette vil gi falske topper. Den morfologiske metoden har en svakhet knyttet til deteksjon av topper som ligger nær områder i matrisen med verdier høyere enn toppen selv. Figur 8.2 illustrerer noen utfordringer knyttet til valg av størrelse på strukturelement. Figuren viser et utsnitt av en radonmatrise, hvor høyeste punkt i området *topp 1* skal detekteres. Et



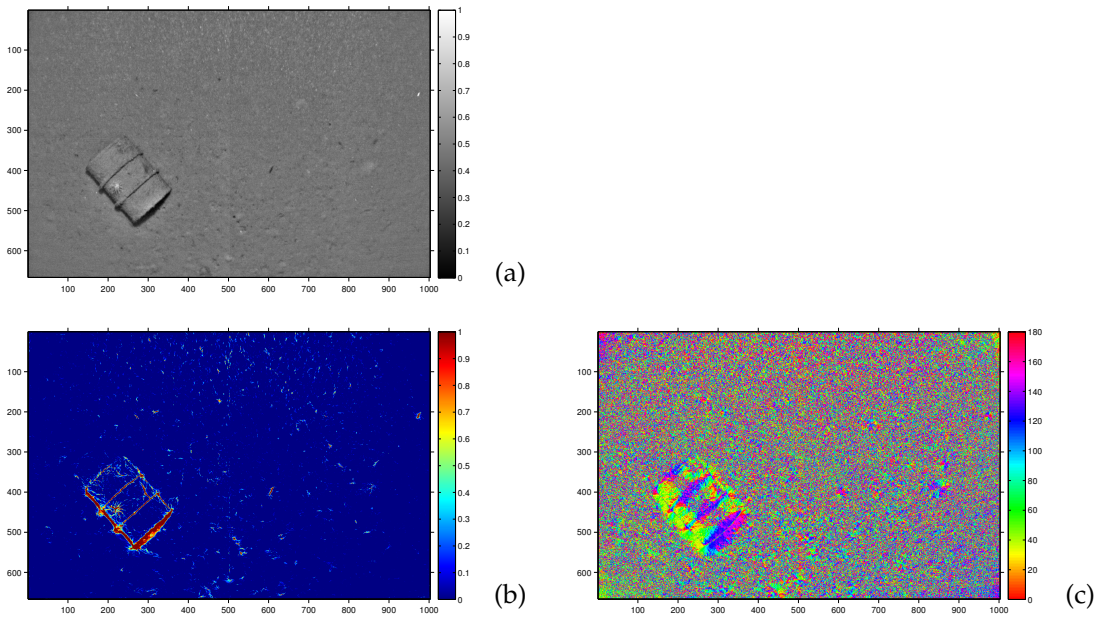
Figur 8.2: *Effekt av ulike størrelser på strukturelement ved deteksjon av lokale maksima i radonmatrisen.*

overdrevent lite strukturelement er markert med de fiolette rektanglene, og som vist kan man få detektert lokale maksima flere steder langs ryggen som danner *topp 1*. Med strukturelementet illustrert med det røde rektanglet har man klart å dekke over et stort nok parti rundt toppen til å unngå dette problemet, mens det grønne rektanglet illustrerer hva som kan skje dersom to topper ligger tett og strukturelementet er for stort. Da vil et vilkårlig punkt fra *topp 2* kunne undertrykke alle punktene i området *topp 1* ved en dilatasjon, og *topp 1* vil ikke detekteres.

Etter inspeksjon av radonmatriser fra bildene i treningssettet ble det valgt et strukturelement med størrelse 20×15 piksler, det vil si et spenn på 20 ρ-verdier og 15 θ-verdier. Linjer som ligger tettere enn 10 piksler fra hverandre (euklidsk avstand) og har forskjell i retning på mindre enn $\pm 7.5°$ vil ikke kunne skilles med dette strukturelementet. Dette ble vurdert som rimelige begrensninger. (For andre bildedimensjoner bør verdiene skaleres). I radonmatrisen i figur 8.1(a) vises også masken omkring hver av de høyeste toppene.

Et annet problem som oppstod i forbindelse med toppdeteksjon, gjaldt topper nær grenseverdiene for θ. Dersom det er valgt å regne ut verdier for θ mellom $-90°$ og $89°$, vil det for en topp ved for eksempel $86°$ kunne oppstå to toppdeteksjoner i radonmatrisen: siden $-90°$ er lik $90°$ og toppene ofte har en viss utstrekning i θ-retning, vil verdiene nær toppen (i eksempelet $86°$) kunne være stigende ned mot $-90°$, og så fortsette å stige i andre enden av matrisen, fra $89°$ til $86°$. Det vil da oppstå lokale maksima både ved $-90°$ og $86°$, og følgelig to topper. Jeg valgte å løse dette ved å la matrisen inneholde overlappende verdier fra $-110°$ til $-109°$, og så deretter bare velge topper som lå mellom $-90°$ og $89°$. I eksempelet vil det da oppstå lokale maksima ved $-93°$ og $86°$, hvorav det ene med sikkerhet kan ekskluderes. I matrisen i figur 8.1(a) er grensene tegnet inn som vertikale linjer.

I oppgaven er det som standard valgt å plukke ut 10 topper, der alle må være høyere enn 40% av høyeste topp.



Figur 8.3: *Homomorf-filtrert undervannsbilde (a), fasekongruens (b) og retning (c).*

8.3 Uttrekking av linjesegmenter langs radonlinjer

For å avgjøre hvor i bildet det finnes faktiske kanter, må de detekterte radonlinjene traverseres piksel for piksel. For det første er man interessert i piksler som indikerer en viss fasekongruens, siden dette betyr at det finnes en kant eller en linje på denne posisjonen i bildet. Man kan anta at kanter tilhørende menneskeskapte objekter består av større grupper av sammenhengende piksler med høy fasekongruens, så små grupper eller enkeltstående piksler kan forkastes. Det kan også være interessant å se på gradientretning i de ulike pikselposisjonene. Fasekongruensfunksjonen kan produsere et slikt retningsbilde, som tilsvarer det man kan regne ut med tradisjonelle gradientoperatorer. Når en radonlinje inspiseres er man i hovedsak interessert i piksler med en retning som stemmer overens med den overordnede linjen. Figur 8.3 viser både fasekongruens og retning for et undervannsbilde. Som figuren viser har pikslene relativt konsistent retning i de områdene av bildet hvor det finnes kanter, mens retningen ser ut til å være mer vilkårlig i områder med lav fasekongruens.

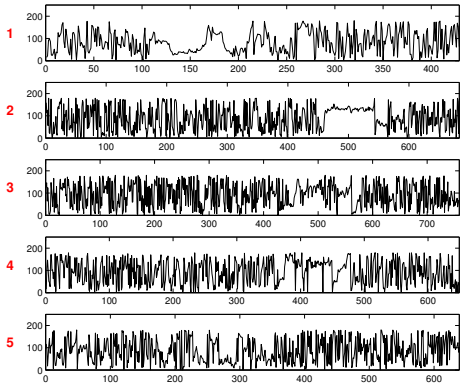
Jeg har prøvd å trekke ut segmenter fra radonlinjene på to forskjellige måter. Begge metodene baserer seg på regiongroing, men «frøene» er valgt ut på forskjellige måter. Etter regiongroing er det lagt til en morfologisk lukking med et 10 piksler langt strukturelement for å slå sammen nærliggende linjesegmenter fra samme radonlinje, men dette er ikke vist i figurene som følger. En nedre begrensning for segmentlengde på 20 piksler er lagt inn som standard for detektoren.

Metode 1 Det første som ble forsøkt, var å se på retningen for alle piksler langs langs radonlinjene. Figur 8.4(a) viser et plott av disse verdiene for radonlinjene som ble vist i figur 8.1(b). Antagelsen var at ekte kanter i bildet har konsistent retning, og dermed vil gi lav varians innenfor et vindu som dekker bare kantpiksler. Jeg valgte et mål på 20 piksler som minimumslengde for interessante kanter, og brukte dette som vindusstørrelse. Resultatet av filtreringen er vist i figur 8.4(b). Filtreringen er gjort med `MATLAB`-funksjonen `colfilt`, som 0-utvider vektoren. De filtrerte linjene er derfor like lange som de opprinnelige. Verdiene er normalisert med hensyn på høyeste verdi, slik at maksimum varians er 1. (En normalisert varians vil være misvisende dersom piksler langs hele radonlinjen skulle ha konsistent retning, men sjansene for dette ble vurdert som svært små).

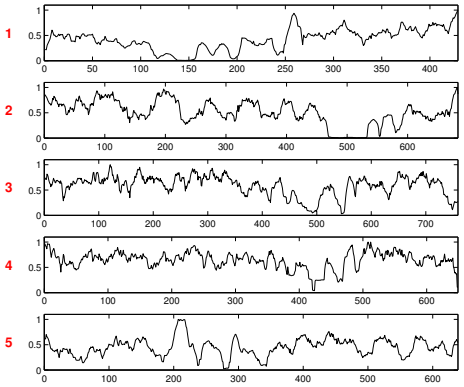
For å detektere linjesegmenter ut fra den filtrerte vektoren, ble den tersklet slik at områder med lav varians ble skilt ut. Det ble brukt hysteresetterskling med verdier 0.03 og 0.05. Terskelverdiene er fastsatt ved eksperimentering. Den tersklede vektoren ble deretter dilatert med et strukturelement av samme størrelsen som ble brukt til variansfiltreringen. Dette ble gjort for å «gjenopprette» deteksjonene til full bredde. Resultatet av terskling med påfølgende dilatering er vist med røde linjer i figur 8.4(d). Disse segmenterte regionene ble brukt som frø for videre regiongroing. Fasekongruens langs radonlinjene er plottet i figur 8.4(c). Fasekongruensverdiene ble også tersklet med hysteresetterskling for å brukes som maske for regiongroingen (terkselverdier [0.7, 0.2]). Maskene er vist med grå felter i (d). Resultatet av regiongroing er vist med sort linje i samme figur.

I dette tilfellet kan vi fra figuren se at ingen frø har blitt plukket ut fra varians langs linje 3 og 4. Ved å sammenligne med linje 2 kan man se at det burde vært deteksjoner omkring *500*, siden de tre linjene er parallelle og kanter svarer til tønnebånd på oljefatet. Siden maskene på sin side så ut til å være fornuftige, ble det forsøkt å bruke en mindre vindusstørrelse for å få flere frø og detektere flere linjesegmenter med variansfiltrering. En vindusstørrelse på 10 piksler ga bedre resultater med de samme terskler for varians (0.05 og 0.03). Resultatet av dette er vist i figur 8.5. Figur (b) viser de detekterte segmentene plottet over det opprinnelige fasekongruensbildet.

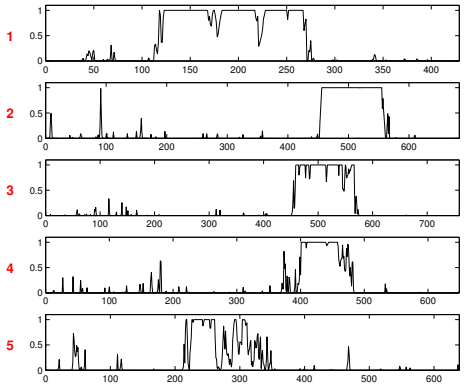
Vertikale kanter For kanter som er tilnærmet vertikale, det vil si at tilhørende radonlinje får en θ-verdi nær 0 eller 180 grader, kan det oppstå et «wraparound»-problem. To piksler med tilnærmet lik retning (180 grader er samme retning som 0 grader) vil kunne ha store forskjeller i verdi, som videre fører til feilaktig høy varians. For å ta hensyn til dette utføres det i aktuelle tilfeller en shift-operasjon på utvalgte retningsverdier langs linjen. Hvis θ ligger mellom 0 og 30 grader, gjøres det en subtraksjon på 180 grader fra alle piksler med retningsverdi over 135. Mulige verdier et piksel kan anta blir da fra -45 til 135. Hvis θ ligger mellom 150 og 180 grader utføres en tilsvarende operasjon, slik at spekteret går fra 45 til 225. Figur 8.6 viser hvilken effekt dette kan ha for en nær vertikal linje. Det finnes en kant langs linjen, omtrent fra verdiene *200* til *400*. Vi kan se hvordan variansen er helt forskjellig i de nederste plottene i figur (c) og (d), henholdsvis uten og med tilpasning av retningsverdier.



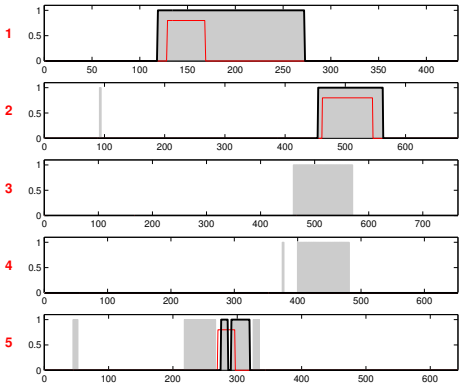
(a) *Retning avlest langs hver radonlinje vist i figur 8.1(b).*



(b) *Varians avlest i et glidende vindu langs linjene i (a). Vindusstørrelse er* 1×20 *piksler.*

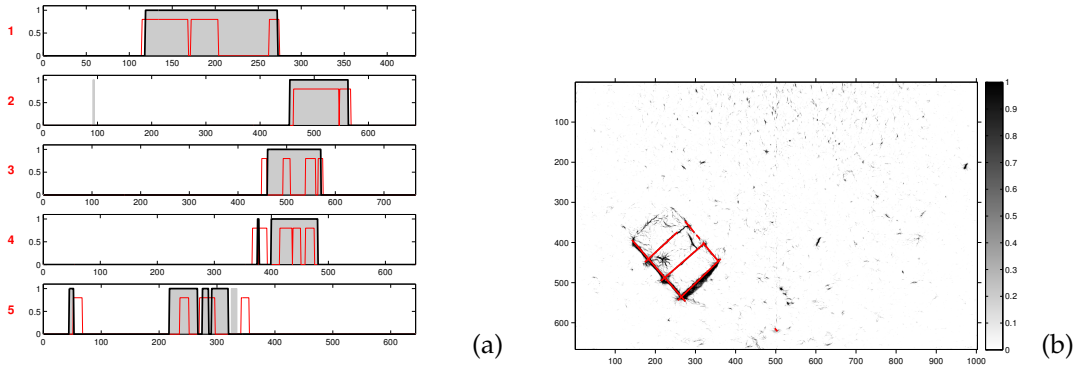


(c) *Fasekongruens avlest langs hver radonlinje.*

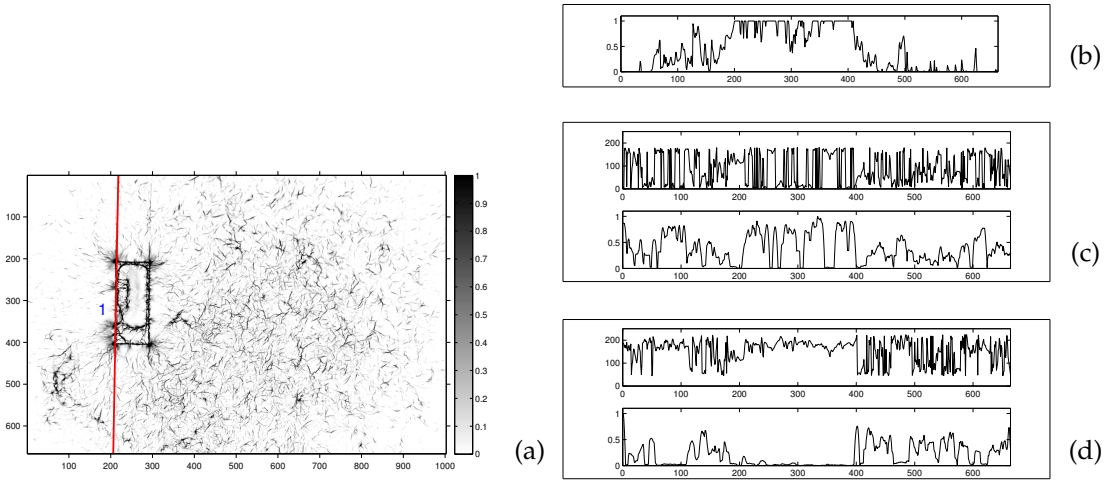


(d) *Rødt (frø): linjene i (b) hysteresetersklet med verdier [0.03, 0.05] og dilatert med vindustørrelse* 1×20*. Grå felt (maske): linjene i (c) hysteresetersklet med verdier [0.7, 0.2]. Sort: Resultat av regiongroing (detekterte segmenter).*

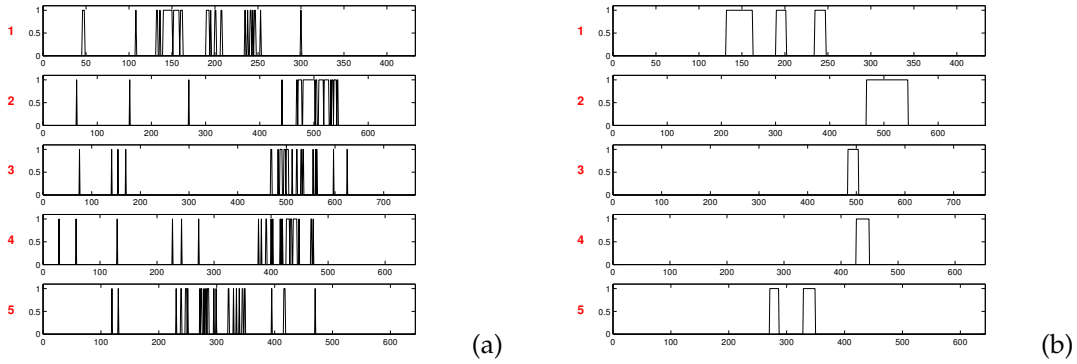
Figur 8.4: *Prosessering av egenskaper avlest langs radonlinjene.*



Figur 8.5: *Segment-deteksjon med mindre vindusstørrelse for variansfiltrering. Fargene i (a) betyr det samme som i figur 8.4(d). De detekterte segmentene er plottet over fasekongruensbildet.*



Figur 8.6: *Wraparound-problem for retningsverdier i nær vertikale linjer. (Originalbilde vist i figur 4.1(h) på side 21.)*
Radonlinje vist over fasekongruensbilde i (a).
Fasekongruens for linjen er vist i (b), kanten tilhørende objektet finnes langs linjen mellom 200 og 400.
I (c) vises retning og varians (hhv. øverst og nederst) før justering av verdier. Verdiene hopper mellom 0 og 180°, noe som fører til høy varians.
Justert retning med tilhørende lav varians er vist i (d).

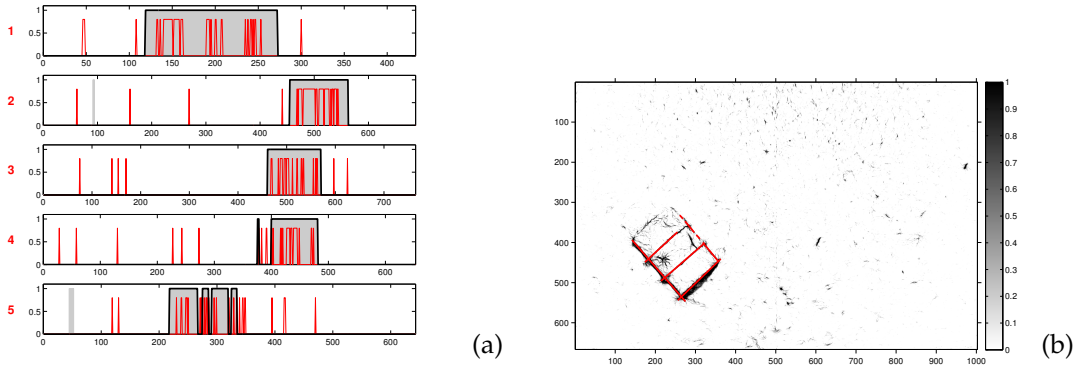


Figur 8.7: *Frøutvelgelse med metode 2. Figur (a) viser piksler med retning lik radonlinjens $\theta \pm 10°$. (b) viser samme signal, filtrert med* `imclose(vector,5)` *og* `imopen(vector,10)` *for fjerning av isolerte frø.*

Metode 2 En slutning som kan trekkes fra resultatene presentert i eksempelet i metode 1, var at masken for regiongroing var ok, men at for få frø ble valgt ut. I den andre metoden ble det også brukt retningspiksler som frø for regiongroing, men disse ble valgt ut ved å se på radonlinjens θ-verdi, og så lete etter piksler med verdi i nærheten av denne. Også her ble det gjort en korreksjon av verdier for vertikale linjer. Som tidligere nevnt, vil piksler uten fasekongruens kunne anta en vilkårlig retningsverdi (se figur 8.3(c) på side 85 for illustrasjon). Det vil dermed kunne dannes flere frø i områder hvor det ikke forventes å finne kanter, så for å begrense disse forekomstene ble alle piksler med lav fasekongruens ignorert. Resultatet av en slik seleksjon er vist i figur 8.7. (I praksis er dette unødvendig, da masken som brukes til regiongroing også stammer fra fasekongruens og stiller krav til en viss magnitude).

Siden det letes etter kanter med en viss utstrekning, ble det forsøkt å foreta en filtrering av vektoren for å fjerne sporadiske treff fra frittstående frø. Ved å bruke morfologisk lukking med et strukturelement av størrelse 1×5 piksler vil frø som ligger tettere enn 5 piksler fra hverandre spleises til ett, stort frø. Med morfologisk åpning med et 1×10 pikslers strukturelement vil deretter alle frø kortere enn 10 piksler fjernes. Som vist i figur 8.7(b) vil dette bidra til å fjerne mange av de uønskede frøene, men som eksempelet også viser fjernes i tillegg flere av de ønskede frøene. Størrelsen på strukturelementene spiller åpenbart en rolle, men det ble valgt å ikke gå videre med denne filtreringen, da motivasjonen for metode 2 var at det ble for få frø med metode 1. Resultatet etter bruk av *ufiltrerte* frø, og maske som i metode 1, er vist i figur 8.8.

Utvidet groing Slik metodene er beskrevet til nå, vil det bare dannes segmenter der masken tillater det. Ved å se tilbake på figur 8.5 kan man imidlertid se at noen frø overlapper og stikker utenfor masken, men ikke blir med i resultatet etter groing. Med et ekstra steg er slike frø inkludert i det grodde resultatet, i håp om å få lengre og mer sammenhengende segmenter. Dette gjøres gjennom følgende steg:



Figur 8.8: *Utvelgelse av linjesegmenter ved bruk av metode 2, uten filtrering av frø-vektoren.*

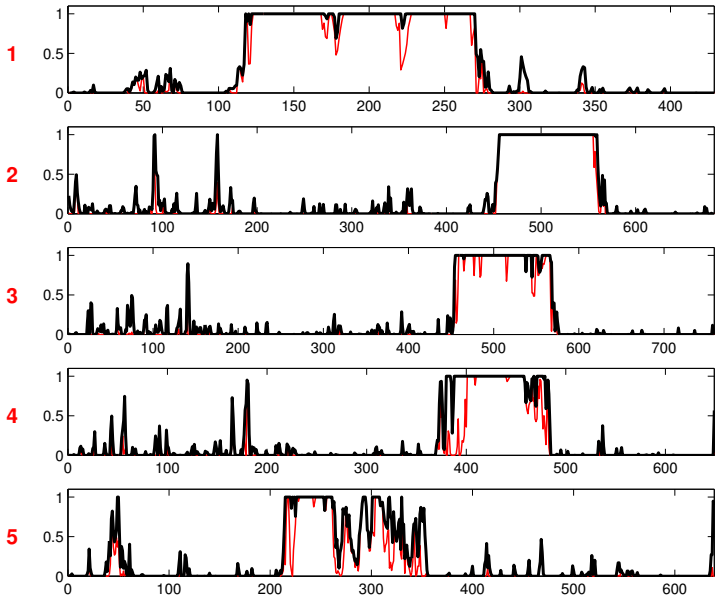
1. frø: `frø`, maske: `maske` $\xrightarrow{\textit{regiongroing}}$ `linjesegmenter1`
2. frø: `linjesegmenter1`, maske: `frø` $\xrightarrow{\textit{regiongroing}}$ `linjesegmenter2`
3. `linjesegmenter = linjesegmenter1 | linjesegmenter2`

Resultatet av denne operasjonen er ikke illustrert med egne figurer.

Utvidet linjebredde Til å begynne med ble det bare trukket ut piksler som lå *direkte under* den detekterte radonlinjen. Senere ble det lagt til funksjonalitet for å også se på et område til sidene for linjen. Et argument for å gjøre dette er at ekte kanter i bildet sjelden vil være helt rette – de kan være svakt buet, objektet kan ha begynt å i oppløsning eller kantene kan se ujevne ut på grunn av sedimenter som legger seg over. Ved å bruke et bredere område omkring radonlinjene kan man forvente å dekke mer av slike kanter. Det vil også gjøre det mulig å sammenligne høyre og venstre side av linjen, noe som kan tenkes å være nyttig for å avdekke hvor et eventuelt objekt ligger i forhold til et oppdaget linjesegment.

Med utgangspunkt i at en linje defineres ut fra ρ og θ, ble områder til sidene for linjen trukket ut ved å variere ρ med $\pm$ {1,2,...} og hente ut pixler for nye linjer på samme måte som for den opprinnelige. Et annet alternativ kunne være å lage en maske som svarer til linjen, for så å dilatere denne og deretter hente ut pikselverdiene under masken. Det best egnede strukturelementet antas da å være en sirkel. En ulempe med denne metoden er at det vil være vanskeligere å implementere en løsning som skiller høyre og venstre side, siden man da må ta hensyn til linjens vinkel.

Fra de utvidede radonlinjene, med lengde L og bredde w piksler, ble det definert en ny egenskap kalt maksimal fasekongruens. For hver rad på tvers av radonlinjen velges pikselet med høyest verdi. Dette gir en ny vektor med lengde L og bredde 1. Linjer samplet på denne måten ble antatt å kunne gi mer sammenhengende kanter. For å hente ut piksler med retning, som beskrevet for metode 1 og 2, vil det finnes flere alternativer. Man kan velge gjennomsnittlig eller median retning på tvers av linjen, eventuelt kan retningsverdien i punktet med høyest fasekongruens velges. Sistnevnte



Figur 8.9: *Maksimal fasekongruens fra utvidet linjebredde (sort) sammenlignet med fasekongruens fra opprinnelig radonline (rødt).*

ble valgt for metode 1. For metode 2 ble det valgt å plukke ut frø fra alle rader som har minst ett piksel med riktig retning. Masken for regiongroing vil ekskludere frø som kommer fra områder med lav fasekongruens.

Det ble eksperimentert med forskjellige linjebredder. Figur 8.9 viser mål på maksimal fasekongruens fra linjebredde 7 (tykk sort linje) plottet over fasekongruens fra den opprinnelige radonlinjen (tynn rød linje). En bredde på 7 vil si at det er lagt til 3 linjer på hver side av den opprinnelige linjen. I dette eksempelet kan det sees at økt linjebredde har ført til at linjesegmenter detektert fra fasekongruens generelt har en mer konsistent høy verdi, siden vi kan se at den røde linjen på flere steder faller nedenfor den sorte. Linjebredde 7 er valgt som standard i oppgaven.

8.4 Gruppering av linjesegmenter til objektdeteksjoner

Ved å velge ut et sett av segmenter slik som beskrevet i forrige avsnitt, er det antatt at disse svarer til faktiske kanter i bildet. Å knytte deteksjoner av menneskeskapte objekter opp mot bare ett enkelt linjesegment vil medføre stor risiko for feildeteksjoner, fordi naturlige forhold og formasjoner på havbunnen også kan danne rette linjer. Når en oversikt over eksisterende kanter er tilgjengelig, kan informasjon om hvordan disse står i forhold til hverandre brukes for å avdekke mulige objekter. De følgende avsnitt

inneholder en detaljert beskrivelse av prosessen fra segmenter til objektdeteksjoner.

8.4.1 Primitiver

Jeg har valgt å definere en type underobjekter som jeg har kalt «primitiver». Dette er grupper av to og to segmenter som er forbundet med hverandre på en måte som kan være sannsynlig for et menneskeskapt objekt, men usannsynlig for kanter som forekommer naturlig på havbunnen. De to primitivene som er valgt for detektoren er parallelle og ortogonale segmenter. Jeg implementerte også en sjekk på om to segmenter fra samme radonlinje kunne tenkes å høre sammen – at de egentlig skulle vært ett langt segment i stedet for to korte – men jeg utelot dette fra det endelige programmet. Å legge til dette primitivet bidro mer til å øke antall feildeteksjoner enn til korrekte deteksjoner, antagelig på grunn av for lave krav og dermed for hyppig spleising. Jeg valgte heller å tillate spleising av segmenter ved å bruke morfologisk lukking på radonlinjene etter terskling og regiongroing.

8.4.2 Parallelle segmenter

Basert på θ-verdier fra radonlinjene ble det laget en liste over alle par av segmenter med tilnærmet lik retning. Jeg valgte å definere to linjer som parallelle dersom forskjell i vinkel var mindre eller lik 3 grader og de stammet fra forskjellige radonlinjer. Grensen for forskjell i vinkel kan endres med brukerparameteren `thetadiff_lim`.

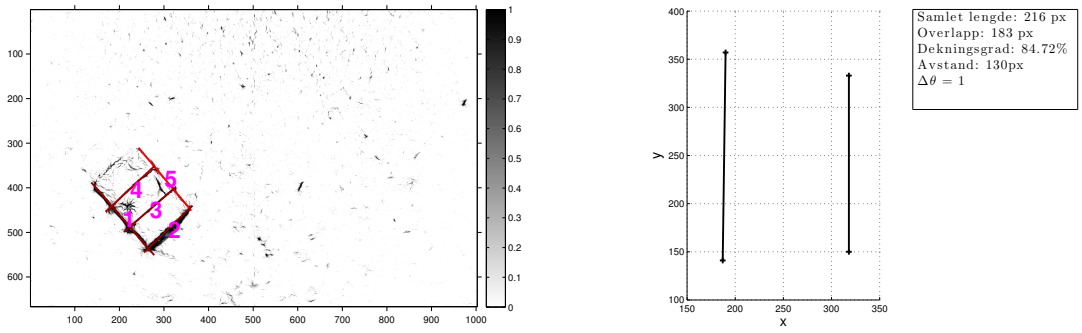
Hvert par av segmenter ble rotert slik at ett av dem stod vertikalt (det andre segmentet vil da også stå vertikalt $\pm 3°$), ved å gjøre en affin transform av segmentenes endepunkter. Punktene ble rotert omkring origo ved å multiplisere med matrisen `T`, gitt av likning (8.1). Origo som brukes i radontransform er definert som `floor((im_dim+1)/2)`, der `im_dim` er bildets dimensjoner.

```
% x0 y0: origo
T  = affineforward(x0,y0,rot_angle);
xy_rot = round(T * [x; y; 1]);
```

$$T = \begin{bmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\,\theta & -sin\,\theta & 0 \\ sin\,\theta & cos\,\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8.1}$$

På denne måten ble det mer lettvint å sammenligne segmentene, siden avstanden mellom dem kunne leses av som en avstand langs x-aksen og graden av overlapp kunne regnes ut fra endepunktenes y-koordinater. Et eksempel er vist i figur 8.10 der segment 1 og 5 sammenlignes. Hvis segmentene overlapper i y-retning ble overlappet kvantifisert ved å dele total lengde utspent av segmentene på lengden med faktisk overlapp. Jeg har kalt denne størrelsen for «dekningsgrad».

Jeg valgte å forkaste parallelle primitiver hvis dekningsgraden var mindre enn 20% eller hvis avstanden mellom segmentene var større enn to ganger lengden av korteste segment. Etter testing av programmet ble det også lagt til en nedre grense for avstand mellom segmenter, slik at primitiver ble forkastet dersom avstanden mellom



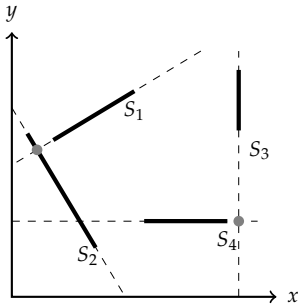
(a) *Detekterte segmenter vist over et objekt* (b) *Segment 1 og 5*

Figur 8.10: *Deteksjon av parallelle primitiver.*

segmentene, det vil si forskjellen i ρ-verdi, var mindre enn 20 piksler. Forekomster av slike parallelle primitiver svarte sjelden eller aldri til deteksjoner av faktiske objekter.

8.4.3 Segmenter som står normalt på hverandre

Som for parallelle segmenter ble det først satt opp en liste over alle par av segmenter som stod tilnærmet normalt på hverandre. For å teste om to segmenter stod nær nok hverandre til å danne et hjørne, fant jeg først skjæringspunktet mellom de to opprinnelige radonlinjene, og deretter segmentenes avstand fra dette punktet. Primitivet ble umiddelbart forkastet hvis minst ett av segmentene var for langt unna. Hvis ikke, ble det gjort tester på hvor skjæringspunktet lå i forhold til segmentene – om de krysset hverandre, om det ene pekte mot det andre eller om de pekte mot et felles punkt. Dersom det var et gap mellom skjæringspunktet og ett eller begge segmentene, gjorde jeg i starten forsøk på å la segmentene gro sammen til et hjørne. Et segment kunne gro mot hjørnet dersom fasekongruensverdier i gapet var over et minimum, og lange segmenter fikk i større grad enn korte lov til å gro over små gap uten fasekongruens. Metoden fungerte greit i endel tilfeller, men jeg valgte etter hvert å utelate denne operasjonen. Dersom det finnes fasekongruens mellom et skjæringspunkt og et segment, kan man fra beskrivelsen i avsnitt 8.3 anta at segmentet på grunn av regiongroing allerede vil strekke seg helt til dette punktet. Det endelige kriteriet for om to segmenter danner et hjørne bestemmes nå bare av avstand til skjæringspunktet. I figur 8.10(a) krysser segmentene hverandre i alle hjørner, og gir sikre primitiver. To andre tilfeller er illustrert grafisk i figur 8.11. Radonlinjene er vist som stiplede linjer, og segmentene som tykke, heltrukne linjer. I dette tilfellet er S_3 langt unna skjæringspunktet som dannes med S_4, så dette primitivet forkastes. S_1 og S_2 står nærmere hverandre og kan danne et hjørne.



Figur 8.11: *Deteksjon av segmenter som står normalt på hverandre. Primitivet* (S_1, S_2) *aksepteres, mens* (S_3, S_4) *forkastes på grunn av avstanden fra* S_3 *til skjæringspunktet.*

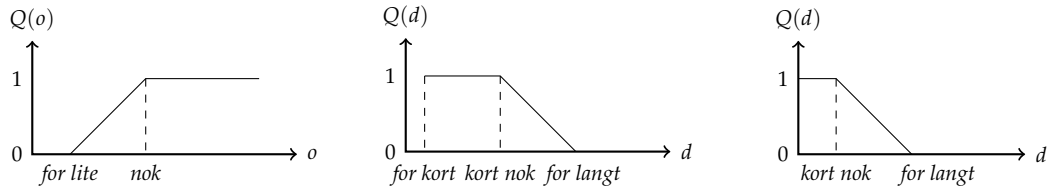
8.4.4 Fra primitiver til objekter

Etter å ha samlet parallelle og ortogonale segmenter til et sett av primitiver, er neste steg å kombinere primitivene til større grupper som representerer én objektdeteksjon. For hvert nye primitiv som kan knyttes til et objekt vil indikasjonene bli sterkere på at objektet kan være menneskeskapt. Man kan si at man samler beviser for at en deteksjon er korrekt, og jo flere beviser jo bedre.

Kombineringen initieres ved å opprette en liste som inneholder alle primitiver, og la hvert primitiv regnes som en «union» av segmenter. Den første unionen i listen sammenlignes så med alle øvrige segmentunioner. Dersom to unioner deler et segment, slås disse sammen til en ny union som legges til i listen, og de to opprinnelige fjernes. Når en union ikke har noe til felles med de andre i listen, lagres den i en egen liste over komplette objektdeteksjoner. Dette gjentas så lenge det finnes elementer i listen over segmentunioner.

8.5 Utvikling av et mål på tiltro til deteksjoner

I dette avsnittet omtales utviklingen av et det jeg har valgt å kalle et «kvalitetsmål», som skal brukes for å skille sikre og usikre deteksjoner. Kvaliteten skal beskrive hvor god deteksjonen er, det vil si hvor sikker man kan være på at deteksjonen svarer til et faktisk menneskeskapt objekt. Kvalitetsverdier omtales i det følgende også som «tiltro» eller «konfidens». Ved kjøring av programmet er det tenkt at brukeren skal kunne sette en terskel for denne kvalitetsverdien, og på denne måten velge hvilke deteksjoner som rapporteres av detektoren. En høy terskel vil bare gi ut deteksjoner som har sterke indikasjoner på å være menneskeskapt. Med en lavere terskel vil vanskeligere objekter kunne detekteres, men da med større fare for at flere av deteksjonene er feil. Egne kvalitetsmål har blitt definert for primitiver og segmenter. Disse brukes til å regne ut en endelig deteksjonskvalitet når flere primitiver samles til én deteksjon.



(a) *Kvalitetsverdi fra dekningsgrad*

(b) *Kvalitetsverdi fra avstand mellom parallelle segmenter*

(c) *Kvalitetsverdi fra avstand mellom ortogonale segmenter*

Figur 8.12: *Grunnlag for utregning av kvalitetsmål for primitiver*

8.5.1 Primitivkvalitet

Jeg ønsket å gi hvert primitiv en score som var avhengig av en tiltro til om det var del av noe menneskeskapt eller ikke. Jeg valgte å knytte denne tiltroen til egenskaper som allerede er beskrevet i de foregående avsnitt – grad av overlapp og avstand mellom parallelle segmenter, og nærhet til skjæringspunkt for ortogonale segmenter. Hvis det settes en enkelt grense for hver av disse verdiene, må man regne med at det alltid vil være tilfeller som akkurat havner på feil side av grensen. Grensen kan settes så høyt at man er sikker på at *ingenting over* den kan regnes som godkjent, men man må da regne med at flere primitiver oppunder grensen burde vært ekskludert. Alternativt kan grensen settes så lavt at *alt under* den må kunne godkjennes, men man risikerer da på tilsvarende måte å forkaste gode primitiver. Man kan forsøke å sette grensen et sted midt i mellom og risikere feil på begge sider av grensen.

Jeg valgte en metode inpirert av såkalt «fuzzy logic» [35], ved å bruke to ytterpunkter som absolutte grenser med score 0 og 1, og så la tilfeller i mellom dem få score et sted mellom 0 og 1, avhengig av avstanden til de to grensene. Fuzzy logic er egnet for problemstillinger med en slik gradvis overgang mellom to (eller flere) tilstander. Det eksemplifiseres gjerne med opplevd temperatur – det føles ikke varmt, men ikke nødvendigvis kaldt heller. Kanskje er den beste beskrivelsen at det er mer kaldt enn varmt, eller motsatt. Et annet eksempel kan være en beholder som er 30% full med vann. Den er hverken tom eller full, men er i dette tilfellet, som eksempelet sier, 30% full, eller 70% tom. Figur 8.12 viser hvordan jeg har implementert en slik logikk for å kvantifisere egenskapene nevnt i de foregående avsnitt. Her er det brukt lineære overganger mellom nedre og øvre grense, men andre funksjoner kunne vært brukt i stedet, dersom det for eksempel var ønskelig med en raskere endring nær en av grensene. En sigmoidfunksjon kunne også vært benyttet. Grensene for «for lite», «nok» etc. ble fastsatt empirisk ved å vurdere segmenter på faktiske objekter i treningssettet. En oversikt over de valgte verdier er vist i tabell 8.1. Her er det brukt verdier målt i piksler, noe som vil være en ulempe for bilder av ulik oppløsning.

For primitiver bestående av parallelle segmenter ble endelig score regnet ut som et gjennomsnitt av kvalitetsmålene fra dekningsgrad og avstand mellom segmentene.

Dekningsgrad for parallelle segmenter	*For lite:* *Nok:*	10 % 60 %
Avstand mellom parallelle segmenter	*For kort:* *Kort nok:* *For langt:*	$\leq$ 20 piksler Lengden av korteste segment 2 $\times$ lengden av korteste segment
Avstand mellom ortogonale segmenter	*Kort nok:* *For langt:*	10 piksler 50 piksler

Tabell 8.1: *Grenser for kvalitetsverdier for primitiver.*

8.5.2 Segmentkvalitet

Det er også laget et kvalitetsmål for segmentene, som skal indikere tiltroen til at segmentet faktisk svarer til en ekte kant. Målet som er definert er ment å være sensitivt for avvik i retning, og er regnet ut som antall piksler med riktig retning (innenfor et slingringsmonn på $\pm 10^\circ$) – delt på segmentets lengde. Denne verdien vektes med gjennomsnittlig fasekongruensverdi. Segmentkvaliteten er videre brukt til å vekte ned primitivkvalitet, slik at et primitiv med gode egenskaper kan få redusert kvalitet dersom det består av segmenter med lav konfidens. Vektingen er gjort ved å ta gjennomsnittet av segmentkvaliteter og primitivkvalitet.

8.5.3 Deteksjonskvalitet

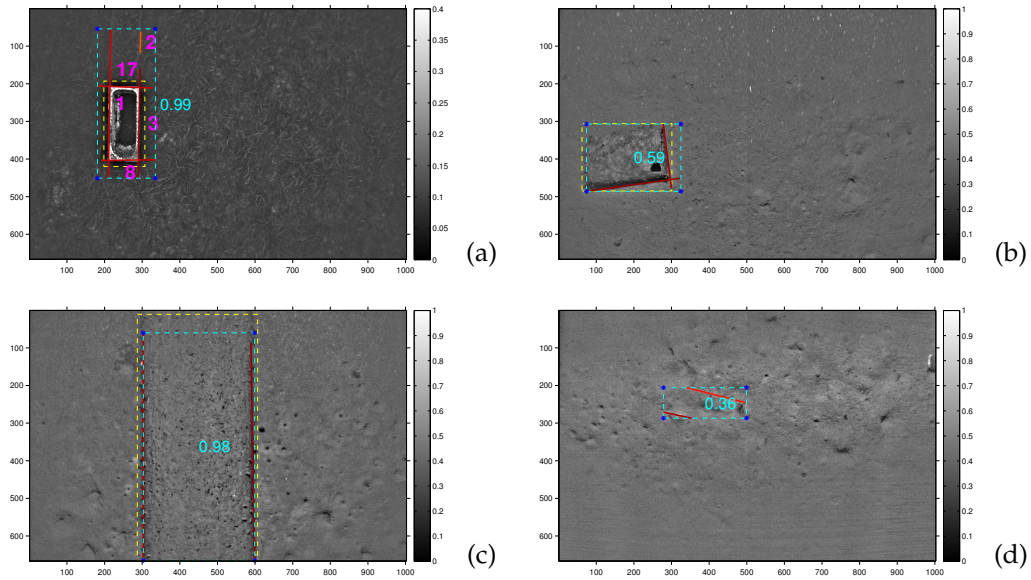
Hovedmålet med å definere en kvalitetsverdi for segmenter og primitiver var å bruke dette til å bygge en endelig deteksjonskvalitet. Tanken var å la kvaliteten til deteksjonen være en funksjon av kvaliteten til dens bestanddeler. Det er naturlig å stole mindre på en deteksjon som bare består av to parallelle segmenter enn på en gruppe som inneholder både flere hjørner og flere paralleller. Et eksempel på en sikker deteksjon er bildet som ble vist i figur 8.10 på side 93. Segmentene danner med hjørner og paralleller totalt ti ulike primitiver som alle kan knyttes sammen til en gruppe gjennom felles segmenter.

En liste med rimelige krav til kvalitetsmålet kan være:

- Kvalitetsverdi mellom 0 og 1, der 1 er best
- Kvaliteten blir bedre jo flere primitiver som knyttes sammen
- Kvaliteten øker raskest i starten:
  Stor forskjell fra ett til to primitiver, liten forskjell fra ni til ti.

Hvis en deteksjon består av n ulike primitiver og man lar P_i være sannsynligheten for at primitiv i er del av et menneskeskapt objekt, vil sannsynligheten for at den samlede deteksjonen er et menneskeskapt objekt være gitt av ligningen

$$P(menneskeskapt) = 1 - (1 - P_1)(1 - P_2)\ldots(1 - P_n), \tag{8.2}$$



Figur 8.13: *Detekterte objekter med utregnet kvalitet. Badekaret i (a) får høy kvalitet fra flere kombinerte primitiver (justert kontrast for trykking). Oljefatet i (b) får nær full score (0.6 for for et ortogonal-primitiv), mens (c) får verdi nær 1 grunnet en svært lang parallell-primitiv. I (d) vises en feildeteksjon som har fått relativt lav kvalitet.*

en ligning som langt på veg tilfredsstiller kravene definert over. Momenter som må nevnes ved bruk av ligningen er for det første at primitivkvalitetene P_i ikke svarer til ekte sannsynligheter, men er egendefinerte verdier. $P(menneskeskapt)$ vil altså heller ikke være en ekte sannsynlighet. For det andre gjelder formelen egentlig for uavhengige sannsynligheter, og i realiteten vil det ikke være rimelig å anta uavhengighet mellom primitivene. For det tredje er en følge av å sette opp uttrykket på denne måten at selv om deteksjonen bare inneholder ett primitiv, kan $P(menneskeskapt)$ bli 1 dersom primitivets kvalitet er 1. Dette vil ikke være helt etter intensjonen, siden en deteksjon med få primitiver bør ha begrenset konfidens. Å bruke denne formelen for deteksjonskvalitet kan med andre ord lett kritiseres, men den ble likevel valgt på grunn av dens enkle logikk.

Utfordringen med høy deteksjonskvalitet ved få primitiver ble løst ved å vekte ned primitivkvalitetene, slik at paralleller kunne få en maksimal score på bare 0.5, mens ortogonale primitiver ble vurdert som en litt sterkere indikasjon og fikk maksimal score 0.6. Et unntak var for parallelle primitiver der overlappet var større enn 75% av bildets høyde. Disse kunne fortsatt oppnå en score på 1, siden parallelle segmenter som går nesten fra kant til kant i bildet vil være en sterk indikasjon på at det har blitt detektert et rør. For slike deteksjoner ble det besluttet at det ikke var nødvendig å kreve flere primitiver for å få full score.

I figur 8.13 vises detekterte objekter og endelig kvalitet på eksempler fra treningssettet. De faktiske tallverdier for utregning av kvalitet for (a), som består av flere primi-

Primitiv	1 2	8 17	1 3	1 8	1 17	3 8	3 17
Kvalitet	0.3005	0.4421	0.4880	0.5591	0.5604	0.5732	0.5744

Tabell 8.2: *Primitivkvaliteter for figur 8.13(a).*

tiver, er vist i tabell 8.2. Dette gir uttrykket

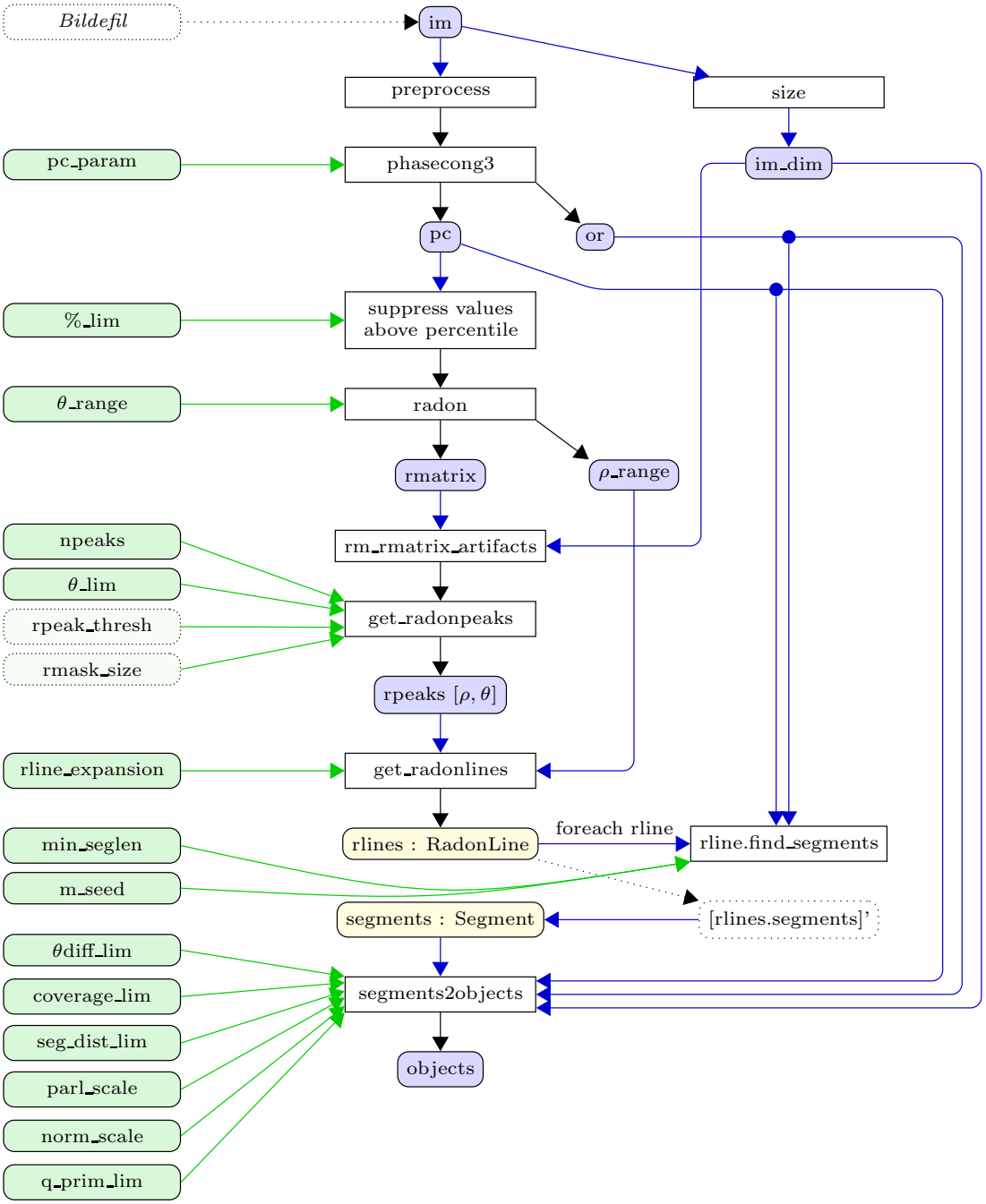
$$\begin{aligned} Q &= 1 - (1 - 0.30)(1 - 0.49)(1 - 0.56)(1 - 0.57)(1 - 0.56)(1 - 0.57)(1 - 0.44) \\ &= 1 - 0.0070 \\ &= 0.9930 \end{aligned}$$

8.6 Programstruktur

Det endelige programmet som er utviklet i denne oppgaven består av en betydelig mengde programkode. Jeg velger å ikke vise eller beskrive noe av denne koden i detalj, men velger i stedet å illustrere hovedtrekkene ved hjelp av et flytskjema i avsnitt 8.6.1. En tabell over parametere som brukeren av systemet kan spesifisere ved kjøring av programmet er vist i avsnitt 8.6.2. For å lettere kunne se gangen i flytskjemaet, gjentas programmets hovedpunkter som ble listet opp i starten av kapittelet:

1. Preprosessere og utføre geometrisk korreksjon
2. Utføre kantdeteksjon ved hjelp av fasekongruens.
   (Dette gir fasekongruensbildet `pc` og retningsbildet `or`).
3. Detektere generelle linjer i bildene med radontransform.
   (Toppdeteksjon `get_radonpeaks` og linjeuttrekking `get_rlines`).
4. Ekstrahere linjesegmenter som svarer til faktiske kanter eller linjer i bildene.
   (`find_segments` for hvert `RadonLine`-objekt).
5. Finne par av linjesegmenter som kan danne hjørner eller paralleller (primitiver).
6. Gruppere primitiver med felles segmenter til én objektdeteksjon.

8.6.1 Flytskjema



Figur 8.14: *Flytskjema for detektoren. Grønne felter indikerer brukerstyrte parametere som kan spesifiseres ved oppstart. Hvite felter er funksjonskall, og blå felter er genererte variable. Gule felter er egendefinerte klasser.*

8.6.2 Opplisting av brukerstyrte parametere

PARAMETER	BESKRIVELSE
Parametere for fasekongruens	
`pc_param`	Parametervalg for fasekongruens. Se avsnitt 7.4.
`prc_lim`	Outlierhåndtering i fasekongruensbildet. Verdier over valgt persentil undertrykkes.
Radontransform	
`theta_range`	Spenn av utregnede θ-verdier (se avsnitt 8.2.1).
`theta_lim`	Gyldige θ-verdier.
`npeaks`	Maksimalt antall topper som plukkes ut.
`rpeak_thresh`	Laveste tillate topphøyde.
`rmask_size`	Maskestørrelse, $\rho \times \theta$.
Uttrekking av linjesegmenter	
`rline_expansion`	Antall linjer lagt til på hver side av radonlinjene.
`min_seglen`	Korteste tillatte linjesegment.
`m_seed`	Metode for frøutvelgelse. (Metode 1 eller 2).
Konstruksjon av primitiver	
`thetadiff_lim`	Maksimalt avvik i retning mellom to linjer for å regne dem som parallelle eller ortogonale ($90° \pm$ grensen).
`coverage_lim`	Nedre og øvre grense for dekningsgrad for parallelle segmenter.
`seg_dist_lim`	Avstand mellom ortogonale segmenter regnest som kort nok, eller for langt til å være et hjørne.
`parl_scale`	Kvalitetsskalering for parallelle primitiver.
`norm_scale`	Kvalitetsskalering for ortogonale primitiver.
`q_prim_lim`	Minste kvalitetsmål som aksepteres for et primitiv (før skalering).

8.7 Verktøy for vurdering av detektorens ytelse

8.7.1 ROC-kurve

En ROC-kurve (*receiver operating characteristics*, eller også *relative operating characteristics*) illustrerer ytelsen til et binært klassifiseringssystem for ulike verdier av systemets kontrollparameter [12].

«Binært» vil si at det finnes bare to mulige klasser som et sampel kan tilordnes, eller sagt med andre ord, at mulige avgjørelser for et detektorsystem er *deteksjon* eller *ikke-deteksjon*. Det er vanlig å velge en kontrollparameter som enten gir ingen falske, eller ingen ekte deteksjoner ved parameterens minimum og maksimumsverdier.

Målet på ytelse er avhengig av to verdier: andel korrekte- og andel falske deteksjoner fra hele testsettet. Andel korrekte deteksjoner, eller *true positive rate* (TPR), er gitt som

$$TPR = \frac{\text{antall korrekte positive deteksjoner}}{\text{antall ekte positive, totalt}}.$$

Andel falske deteksjoner, *false positive rate* (FPR), måles på en tilsvarende måte:

$$FPR = \frac{\text{antall falske positive deteksjoner}}{\text{antall ekte negative, totalt}}.$$

Som et konkret eksempel på de to målene, kan man tenke seg at det skal stilles en sykdomsdiagnose. TPR vil da være antall syke personer som har fått diagnosen syk – delt på det totale antall syke personer, mens FPR vil være antall *friske* personer som har fått diagnosen syk – delt på totalt antall friske personer. Systemets ytelse er gitt av forholdet $\frac{TPR}{FPR}$. Resultatet for én valgt verdi av kontrollparameteren plottes som et punkt i et koordinatsystem med TPR langs y-aksen og FPR langs x-aksen. Slik regnes ytelse ut for et bredt spekter av parameterverdier, og en kurve trekkes gjennom de plottede punktene. Dette er ROC-kurven, som sier noe om systemets evne til å skille mellom de to klassene ved ulike verdier for kontrollparameteren.

For økende verdier av denne parameteren vil til å begynne med feilraten typisk reduseres uten at det går særlig på bekostning av antallet korrekte deteksjoner. Dette fortsetter opp til et visst punkt, da også TPR begynner å falle. Dette punktet med minimal FPR og maksimal TPR kalles gjerne for et «kne». Når kontrollparameteren økes ytterligere går typisk både TPR og FPR mot 0.

I denne oppgaven vil mulige klasser være «menneskeskapt» og «ikke-menneskeskapt», og et sampel vil inneholde en verdi som indikerer hvilken tiltro programmet har til at et detektert objektet er menneskeskapt. Antallet ikke-menneskeskapte (ekte negative) objekter i datasettet har ingen åpenbar verdi i dette tilfellet. Jeg har valgt å dele antall feildeteksjoner på antall bilder, slik at FPR gir gjennomsnittlig antall feildeteksjoner pr. bilde. Kvalitetsverdi for objekter er brukt som kontrollparameter.

8.7.2 Lage en fasit og teste detektoren

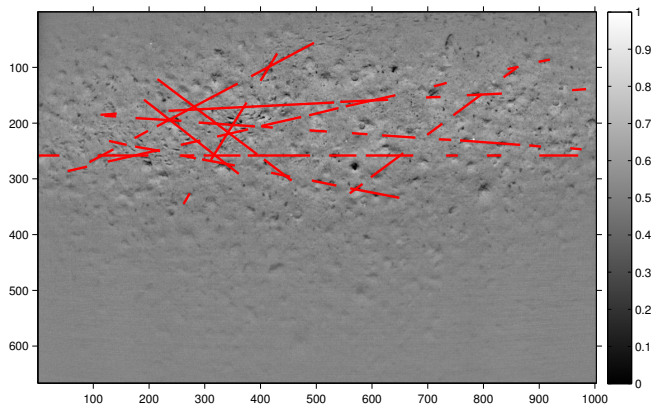
For å vurdere detektoren lages det en fasit av datasettet som detektoren skal testes på. Ved testing blir ett og ett bilde inspisert, og hver deteksjon sammenlignes med hvert fasitobjekt. For at en deteksjon skal regnes som riktig, har jeg valgt å stille som krav at alle segmenter må være på objektet. På denne måten gjøres det en opptelling av antall ekte- og falske positive deteksjoner, og ved hvilke verdier av brukerparameteren (deteksjonskvalitet) et fasitobjekt kunne detekteres. Resultatene blir presentert som ROC-kurver.

8.7.3 Vurderinger for valg av fasit

Når det lages en fasit, må det gjøres et valg for hvilke typer objekter som skal inkluderes. Hvis målet er å teste evnen til å finne alle menneskeskapte objekter, uavhengig av form og karakter, må alle avbildede objekter registreres. Med modellen som er valgt i denne oppgaven – å lete etter parallelle og ortogonale kanter – kan det ikke forventes at alle objekter blir detektert, og TPR vil sannsynligvis ikke kunne nå helt opp til 1. Det andre alternativet er å bare inkludere i fasiten objekter som faktisk har parallelle eller vinkelrette kanter, og som dermed er teoretisk mulig å finne ved hjelp av de valgte primitiver. Dette vil være mest «rettferdig» i forhold å vurdere hvor god detektoren er til å finne akkurat den typen objekter den er laget for å finne. Hvis den på slump skulle finne menneskeskapte objekter som ikke svarer til modellen, må disse da regnes som falske deteksjoner. I et generelt tilfelle vil det være interessant å se på ytelsen til detektoren fra begge perspektiver, men blant bildene som er valgt ut er det bare et fåtall som klart skiller seg ut som ikke detekterbare (ved hjelp av rette linjer). Derfor er det valgt å medregne alle menneskeskapte objekter i fasit for trening- og testsett.

8.7.4 Vurderinger for godkjente deteksjoner

Det må også tas stilling til hvordan man skal håndtere tilfeller der falske deteksjoner – det vil si deteksjoner som ikke svarer til objektenes faktiske kanter – har havnet over objektene. Ved bruk av detektorprogammet i praksis er det selvfølgelig en fordel at så mange objekter som mulig detekteres, men ved evaluering av ytelse det må gjøres en vurdering for hvordan slike sporadiske forekomster skal håndteres. Om disse regnes som ekte eller falske positive er et definisjonsspørsmål som vil påvirke utregningen av ROC-kurven og dermed målet på detektorens ytelse. I oppgaven er det valgt å la deteksjoner som er helt omsluttet av det faktiske objektet telle som ekte, mens deteksjoner der bare ett linjesegment ligger på eller innenfor objektet blir regnet som falske.



Figur 8.15: *Falske kantdeteksjoner i bilder av mudderbunn med noe struktur.*

8.8 Valg av parametere for detektoren

Gjennom de foregående kapitler og avsnitt har det blitt vist metoder, eksperimenter og delresultater som steg for steg har ledet frem til et endelig detektorprogram. Underveis i utviklingen av detektoren har bildene i treningssettet blitt brukt til å vurdere ytelsen, og det er det lagt inn forslag til verdi for alle parametere som ble listet opp i avsnitt 8.6.2. Målet har vært å gjøre gode objektdeteksjoner uten for mange feil, men likevel har det vist seg å være noen utfordringer knyttet til en spesiell type bilder i treningssettet. Detektoren er tilsynelatende svak for bilder som ikke har noen tydelige kanter, for eksempel mudderbunn med bare noe struktur i form av huller og groper. For slike bilder blir det ofte dannet en rekke segmenter på tvers av bildet og segmentene danner ofte parallelle primitiver. Et eksempel er vist i figur 8.15, der de røde linjene illustrerer detekterte kanter. Det vil kunne hjelpe å trekke ut færre topper fra radonmatrisen og således få færre linjesegmenter totalt, men dette har ikke blitt vurdert som en aktuell løsning fordi en reduksjon av antall radonlinjer vil gjøre det vanskeligere (eller umulig) å detektere flere objekter i samme bilde. Det vil også være uheldig for objekter med mange kanter (eksempelvis oljefatet), da det blir vanskeligere å oppnå en høy deteksjonskvalitet.

I dette avsnittet beskrives noen alternativer som prøves ut for å redusere problemet. Det presiseres at eksperimenteringen utføres på *treningssettet*, som består av 22 bilder. Basert på eksperimenteringen gjøres det et valg for hvilke brukerstyrte parametere som egner seg best for detektoren. Dette settet av parametere vil brukes til slutt når detektorens ytelse skal måles på et uavhengig *testsett*.

8.8.1 Eksperimentering på treningssettet

Alternativene som prøves er:

0. Initiell utprøving av detektoren

1. Økt støyfiltrering i fasekongruensfilteret
2. Alternativt valg av frømetode, med og uten støyfiltrering
3. Deteksjon uten bruk av utvidet linjebredde

Eksperiment 0 - Initiell utprøving av detektoren

Gjennom utviklingen av detektoren er det lagt inn forslag til verdi for alle parametere som ble listet opp i avsnitt 8.6.2, og dette har ledet frem til en initiell utprøving. Av de parametere som kan ha innvirkning på problemet som ønskes løst med denne eksperimenteringen, er de viktigste listet opp under:

- `'pc_param'` som beskrevet i avsnitt 7.4.
- `'npeaks'`, antall topper/linjer, er satt til 10
- `'rline_expansion'` er satt til 3 (linjebredde 7, 1+3+3)
- `'m_seed'` frømetode 2 er valgt for regiongroing (se avsnitt 8.3)

Eksperiment 1 - Økt støyfiltrering i fasekongruensfilteret

En mulig løsning for å redusere problemet er å justere faskongruensparametrene til kraftigere støydempning. De fleste av objektene i treningssettet har relativt brede kanter, så det er antatt at en justering av parametere med hensyn på å fjerne de mest høyfrekvente bidragene (se eksperimentering i avsnitt 7.4) kan hjelpe. Å justere kravet til spredning i frekvens er også lagt inn for å få færre feildeteksjoner på slett havbunn. Følgende endringer er gjort i parametrene til fasekongruensfilteret:

- `'minwavelength'` fra 2 til 4 – for å fjerne større spekter av høyfrekvent støy
- `'cutoff'` fra 0.25 til 0.45, og
- `'g'` fra 5 til 8 for å fjerne mer av de aller mest lavfrekvente bidragene.

For beskrivelse av parametrene henvises det til avsnitt 7.4.

Eksperiment 2 - Frømetode 1

For å fjerne de uønskede segmentene som oppstår, testes bruk av frømetode 1 som ser på lokal varians langs radonlinjene (se avsnitt 8.3). Dette kan styres med brukerparameteren `'m_seed'`. Hypotesen er at det vil dannes færre frø på grunn av færre områder med konsistent retning i områder med bare havbunn, og dermed færre ukorrekte segmenter. Eksperimentering med frømetode 1 gjøres i to varianter:

2a Med støyfiltreringen som ble introdusert i «eksperiment 1», for å se den kombinerte effekten,

2b Uten støyfiltrering, for å se om frømetode alene kan bedre resultatet.

Eksperiment 3 - Smale radonlinjer

Det er antatt at økt linjebredde for radonlinjene gir mer sammenhengende kanter, så bredde 7 er brukt som standard. For å teste antagelsen gjøres det i dette eksperimentet forsøk med linjebredde 1. En teori er at smalere linjer gjør det vanskeligere for falske segmenter å gro seg lange. Linjebredden endres ved å sette parameteren `'rline_expansion'` til 0. Frømetoden er i dette eksperimentet igjen satt til 2.

8.8.2 Inspeksjon av resultater fra eksperimentering og valg av parametere

Inspeksjon av resultater Det viser seg å være relativt stor variasjon i resultatene fra de ulike eksperimentene. For enkel sammenligning er ROC-kurvene vist i figur 8.16. Den initielle testen (figur (a)) gir generelt høy feilrate, og dersom alle deteksjoner aksepteres vil det i gjennomsnitt svare til mer enn 1 feildeteksjon pr bilde. Et slikt resultat er ikke overraskende, da dette antas å stamme fra «sporadiske» segmenter som vist i figur 8.15, og som i hovedsak var motivasjonen for de utførte eksperimentene. I punktet 0.5 er $TPR = 0.87$ og $FPR = 0.32$, som er laveste feilrate for flest mulig deteksjoner.

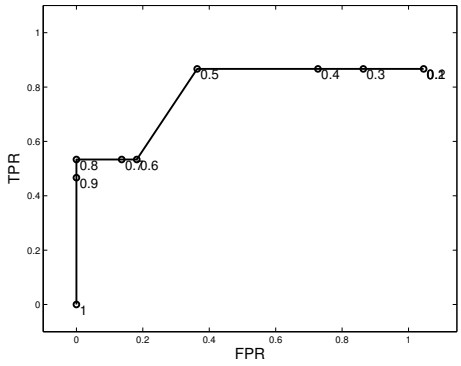
Med økt støyfiltrering viser resultatet i (b) stor forbedring, dog på bekostning av antall korrekte deteksjoner. Her er det et enda tydeligere knekkpunkt ved 0.5. I dette punktet er $TPR = 0.67$ og $FPR = 0.05$, en stor forbedring av FPR.

Bruk av frømetode 1, vist i (c) og (d) reduserer også antall feildeteksjoner. Med støyfiltrering faller antallet korrekte deteksjoner ytteligere og feilraten øker sammenlignet med eksperiment 1, hvis kontrollparameterverdi 0.5 brukes til sammenligning. Uten støyfiltrering er resultatet mer lovende, med relativt få feil og en rate for korrekte deteksjoner som er like høy som i den initielle utprøvingen. I punktet 0.5 er $TPR = 0.87$ og $FPR = 0.18$.

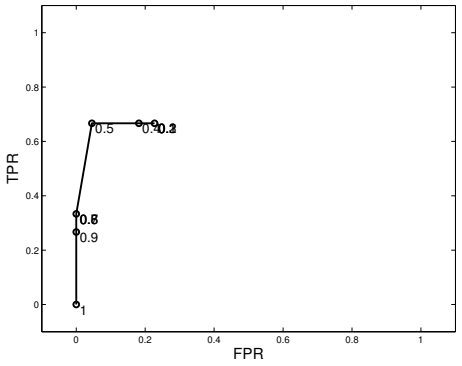
Som forventet ved bruk av smalere radonlinjer, ble det observert mindre sammenhengende kanter og generelt færre detekterte linjesegmenter. Dette resulterte i færre deteksjoner både på og utenfor fasitobjektene, men det bør nevnes at de segmentene som *ble* trukket ut i all hovedsak svarte til faktiske kanter. Det er ikke tatt med figurer som viser dette i rapporten. ROC-kurven i (e) viser relativt lave forekomster av både ekte og falske positiver, og ellers at ytelsen er svært lik den fra eksperiment 2a. For redusert linjebredde antas det å være nødvendig med nye vurderinger i forhold til regiongroing og krav til primitiver for å detektere flere objekter.

Valg av parametere Hensikten med eksperimenteringen har vært å finne et sett av parametere som gir lav feilrate og samtidig flest mulig korrekte deteksjoner på treningssettet. Med det utgangspunktet som er gitt i form av mulige parametervalg, vurderes frømetode 1 (lokal varians) til å være det beste alternativet, sammen med initielle valg for fasekongruensparametere (ikke ytterligere utvidet støydemping).

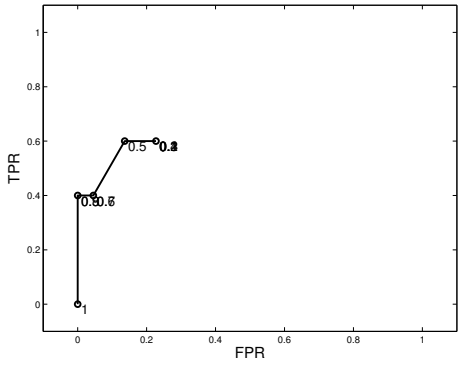
Frømetode 1 velges fordi det tilsynelatende er et forbedringspotensiale i steget fra radonlinjer til linjesegmenter, og denne metoden stiller strengest krav til konsistent retning og dermed best unngår uttrekking av falske linjesegmenter. Videre er et fellestrekk for ROC-kurvene i figur 8.16 at FPR nærmer seg sitt minimum ved



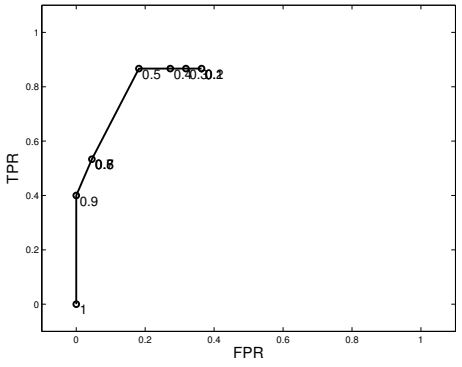
(a) *Eksperiment 0, initiell utprøving*



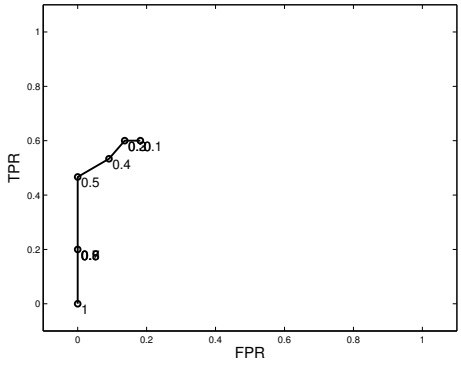
(b) *Eksperiment 1, økt støyfiltrering*



(c) *Eksperiment 2a, frømetode 1 og støyfiltrering*



(d) *Eksperiment 2b, frømetode 1 uten støyfiltrering*



(e) *Eksperiment 3, redusert linjebredde*

Figur 8.16: *ROC-kurver fra eksperimentering på treningssettet.*

kontrollparameterverdi (objektkvalitet) 0.5, som vil si at komplekse deteksjoner bestående av mange primitiver sjelden svarer til falske deteksjoner. Derfor kan det argumenteres for at støyfiltrering utelates, slik at så få reelle kanter som mulig fjernes. Det bør forøvrig nevnes at med et så lite treningssett (22 bilder) vil én korrekt deteksjon fra eller til kunne ha en merkbar påvirkning på ROC-kurven.

8.9 Testoppsett

Med bakgrunn i eksperimentene fra avsnitt 8.8 og utviklingen av detektoren forøvrig, er det gjort et valg for endelige verdier til detektorens brukerstyrte parametere. Dette valget av parametere vil brukes når detektorens ytelse skal evalueres på testsettet. Tabell 8.3 viser de valgte verdier. For beskrivelse av parametrene henvises det til avsnitt 8.6.2 på side 100.

Forventet ytelse fra inspeksjon av testsettet For en karakteristikk av bildene henvises det til kapittel 4, da beskrivelsen der er generell for både trening- og testsett. Testsettet består av 157 relativt varierte bilder. Av disse er det 83 som inneholder ett eller flere menneskeskapte objekter, med totalt 92 objekter markert i fasiten. Utvalget er gjort uten å ta hensyn til hverken bildekvalitet eller modell for menneskeskapte objekter, så på grunn av form, perspektiv eller dårlig kontrast anslås det at rundt 15 stykker ikke vil detekteres. Med dette anslaget forventes det en TPR på 0.84 eller lavere, altså høyst 77 deteksjoner av 92 mulige.

Tilsvarende er det anslått at om lag 20 bilder inneholder naturlige objekter eller havbunn som kan gi feildeteksjoner, og i bilder med for eksempel oppsprukken berggrunn med rette bruddflater vil det fort kunne oppstå både to og tre feildeteksjoner. Som en løs antagelse forventes det rundt 30 feildeteksjoner, som gir en FPR på 0.19. Flere av de antatte feildeteksjonene vil kunne skilles ut ved å stille krav til flere primitiver og dermed høy deteksjonskonfidens, men dette vil også ramme ekte objekter som detekteres med for eksempel bare én parallell. Estimatet på 0.19 baserer seg bare på en opptelling av objekter og havbunnsformasjoner med kanter som potensielt kan lede til falske deteksjoner. På grunn av de tidligere omtalte problemene med «strøsegmenter» fra detektoren er ikke estimatet særlig realistisk, så i praksis forventes det en god del flere feil.

PARAMETER	***VERDI***
`pc_param`	*Se avsnitt 7.4 på side 72.*
`prc_lim`	*99.5*
`theta_range`	*[-110:109]*
`theta_lim`	*[-90, 89]*
`npeaks`	*10*
`rpeak_thresh`	*(40% av max)*
`rmask_size`	*[20, 15]*
`rline_expansion`	*3*
`min_seglen`	*20*
`m_seed`	*1*
`thetadiff_lim`	*3*
`coverage_lim`	*[10, 60]*
`seg_dist_lim`	*[10, 50]*
`parl_scale`	*0.5*
`norm_scale`	*0.6*
`q_prim_lim`	*0.2*

Tabell 8.3: *Parameterinnstillinger for detektoren brukt til testing på testsettet.*

Kapittel 9

Resultater

I dette kapittelet presenteres og diskuteres resultatene fra testingen.

9.1 Testresultater

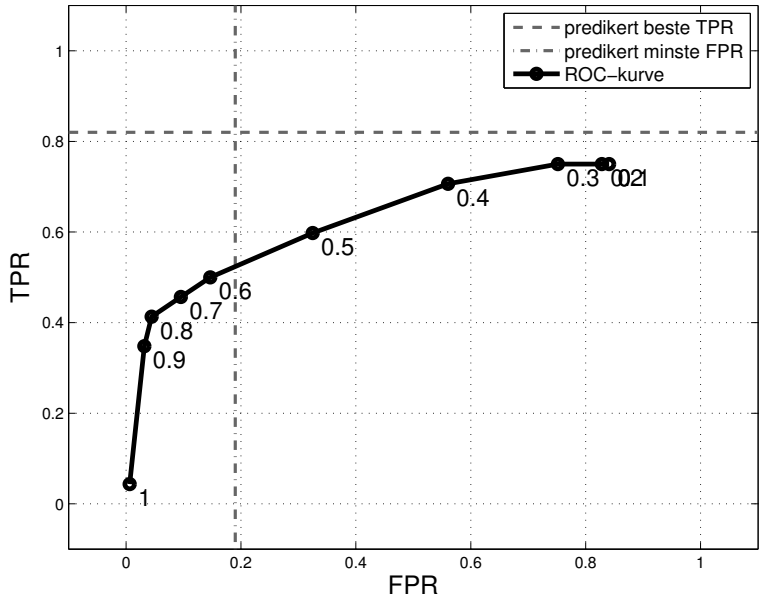
Resultatene av testen som ble beskrevet i avsnitt 8.9 er vist ved hjelp av ROC-kurven i figur 9.1. For ekstra tydelighet er TPR og FPR-punktene i kurven listet opp i tabell 9.1 sammen med en oversikt over faktiske antall korrekte og falske deteksjoner.

9.1.1 ROC-kurve

Sammenligning med predikert ytelse Etter en inspeksjon av testsettet ble det i avsnitt 8.9 fastslått at detektoren sannsynligvis ikke kunne finne alle objekter markert i fasiten, så et estimat for beste TPR ble satt til 0.82. Dersom alle deteksjoner aksepteres, det vil si at kontrollparameteren eller terskelverdien for deteksjonskvalitet er 0, ser vi at TPR ligger tett opp mot forventet ytelse. Detektoren var altså i stand til å oppdage omtrent like mange objekter som forventet, men bare ved terskelverdier som samtidig gir relativt høy feilrate.

Sammenligning med predikert feilrate Det ble også anslått et sannsynlig antall feildeteksjoner basert på bilder med utfordrende havbunn og naturlige objekter, som ga en predikert feilrate FPR på 0.19. I ROC-kurven kan vi se at denne verdien oppnås hvis deteksjonskvalitetsparameteren er mellom 0.5 og 0.6. Dette virker fornuftig, siden disse verdiene er beste mulige kvalitet for henholdsvis parallelle og ortogonale enkeltprimitiver. Resultatet sier oss at krav til mer komplekse deteksjoner (grupper av primitiver) vil kunne sile ut de *forventede* feildeteksjonene i dette datasettet.

En kommentar til at resultatet er tilnærmet likt for tersklene 0 til 0.2 er at en hardkodet grense i programmet gjør at primitiver med kvalitet under 0.2 forkastes. En vekting med segmentkvalitet gjør at noen objekter likevel får lavere kvalitet.



Figur 9.1: *Detektorens ytelse på testsettet illustrert med ROC-kurve. Horisontal stiplet linje illustrerer forventet beste mulige resultat (0.84). Vertikal stiplet linje viser forventet antall feildeteksjoner totalt, basert på vanskelige eller ikke menneskeskapte objekter. For nøyaktige verdier, se tabell 9.1.*

Kne-punkt for ROC-kurven Basert på kurvens form, vil kneet intuitivt defineres ved terskelverdi 0.8. Den store forskjellen før og etter denne terskelen er endringen i TPR, som stuper hvis terskelen heves. Et optimalt knepunkt bør egentlig settes slik at feilraten FPR er redusert så mye som mulig uten at det har gått nevneverdig på bekostning av TPR, men for denne kurven ser vi at TPR faller jevnt og trutt allerede fra terskelverdi 0.3, at og et kne ved 0.8 vil ekskludere så mye som en tredjedel av alle korrekte deteksjoner.

Kurvens form og antall deteksjoner En annen interessant observasjon er at forholdet mellom TPR og FPR reduseres nærmest lineært i området terskel 0.3 til 0.6. Med det menes det at en endring i TPR ser ut til å være tilnærmet proporsjonal med endringen i FPR, og en slutning er at mange både ekte og falske deteksjoner består av bare ett primitiv. Dette støttes av tallverdiene i tabell 9.1, der vi ser at 46 objekter, halvparten av antallet markert i fasiten, har kvalitet 0.6 eller bedre, som impliserer at den andre halvparten har kvalitet lavere enn 0.6 eller ikke blir detektert.

Den mest åpenbare konklusjonen er at det oppstår for mange falske primitiver. Grupper av primitiver (kvalitet bedre enn 0.6) gir *relativt* lav feilrate og er i seg selv et godt mål på «menneskeskapthet», men som sagt gjelder dette bare for halvparten av objektene i fasiten. Et problem er altså at detektoren ikke kan diskriminere mellom ekte og falske deteksjoner som består av bare ett primitiv. Inspeksjon av tabellen for terskel

Terskel	*TPR*	*FPR*	Antall korrekt	Antall feil
0	0.75	0.84	69	132
0.1	0.75	0.84	69	132
0.2	0.75	0.83	69	130
0.3	0.75	0.75	69	118
0.4	0.71	0.56	65	88
0.5	0.60	0.32	55	51
0.6	0.50	0.15	46	23
0.7	0.46	0.10	42	15
0.8	0.41	0.04	38	7
0.9	0.35	0.03	32	5
1	0.04	0.01	4	1

Tabell 9.1: *Detektorens ytelse på testsettet. For en gitt terskelverdi medregnes alle deteksjoner med en konfidens større eller lik terskelverdien. Tabellen viser verdiene både i form av TPR og FPR, og som faktisk antall deteksjoner. TPR er antall korrekte deteksjoner delt på totalt antall korrekte (92), mens FPR er antall feildeteksjoner pr bilde, dvs. antall feil delt på 157 bilder.*

0.6 viser 46 korrekte og 23 falske deteksjoner. Dette gir forholdet 2:1, som tilsier at hver tredje deteksjon er feil.

9.1.2 Inspeksjon av eksempelbilder fra testsettet

I dette avsnittet vises endel deteksjoner i bilder fra testsettet. I bildene er fasitobjekter markert med gule rektangler med stiplede linjer. (Stort sett, men ikke alltid, er hele objektet dekket av dette rektangelet.) I bildene er alle deteksjoner vist, uavhengig av kvalitet. Dette vil svare til verdi 0 for kontrollparameteren i ROC-kurven. Eventuelle deteksjoner er markert med lyseblå rektangler, samt tiltro til deteksjonen markert med lyseblå tall. Merk at noen bilder har komprimert gråtoneskala fra originalt [0, 1] for bedre visning på papir.

I figur 9.2 vises deteksjoner av objekter som stemmer godt over ens med modellen. Flere sett av paralleller og ortogonaler gir høy deteksjonskonfidens. Bildene har generelt god kontrast og tydelige kanter på objektene. For deteksjonene i (g) og (h) svarer ikke alle segmenter til åpenbare kanter, men har likevel bidratt til å gi gode deteksjoner totalt sett.

Figur 9.3 viser resultater fra et utvalg rørbilder. Her har detektoren gjort flere gode deteksjoner. I (d) er det ingen paralleller, men på grunn av klossen midt på røret blir det to ortogonaler som til sammen dekker mye av røret. I (g) og (h) gjøres det flere

uavhengige deteksjoner langs røret.

I figur 9.4 vises flere typer objekter, også her med flere greie deteksjoner. Noen av bildene inneholder feildeteksjoner, men felles for disse er at de generelt har lav konfidens og kan lukes ut ved å stille krav til en viss objektkvalitet. I (c) er objektet detektert med tre ulike deteksjoner som ligger over hverandre. Fra dette kan man tenke seg at det kunne være aktuelt å slå sammen deteksjoner selv om de ikke knyttes sammen ved hjelp av felles segmenter.

Til slutt er det vist noen feildeteksjoner i figur 9.5. Her er det tydelig at detektoren fortsatt har et stort potensiale når det gjelder uttrekking av segmenter. I hverken (a), (b) eller (c) kan segmentene sies å stamme fra konkrete kanter i bildene. I (a) finnes det vertikale kanter midt i bildet, men likevel er det ute på havbunnen til høyre at det oppstår deteksjoner. Med bildet som er vist i (d) (maneten *Periphylla Periphylla*) ser man at selv med gode segmenter som svarer til faktiske linjer er det ikke sikkert at sikre deteksjoner alltid svarer til menneskeskapte objekter.

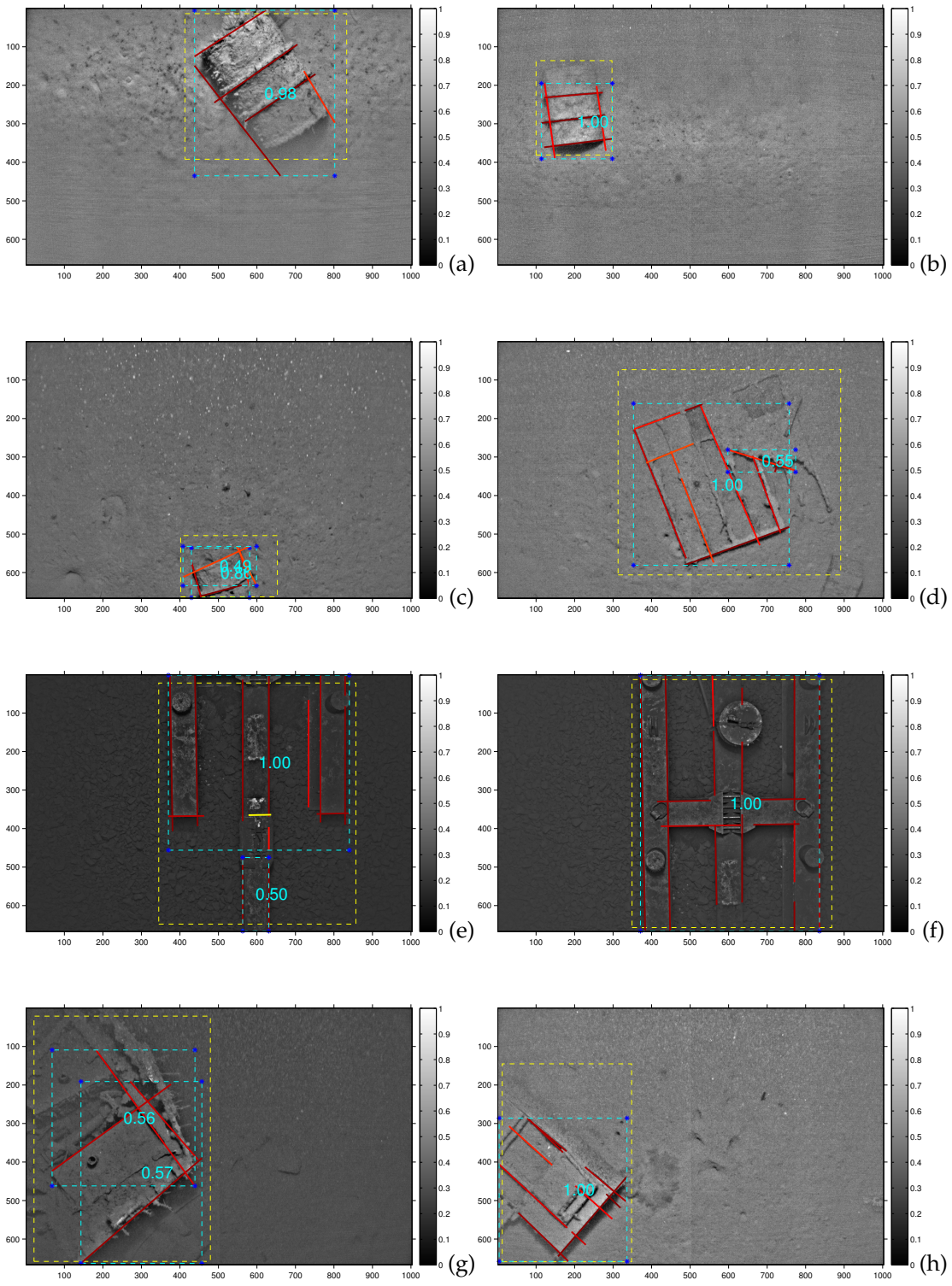
9.1.3 Oppsummering av testresultater

Oppsummering fra ROC-kurve og tabell

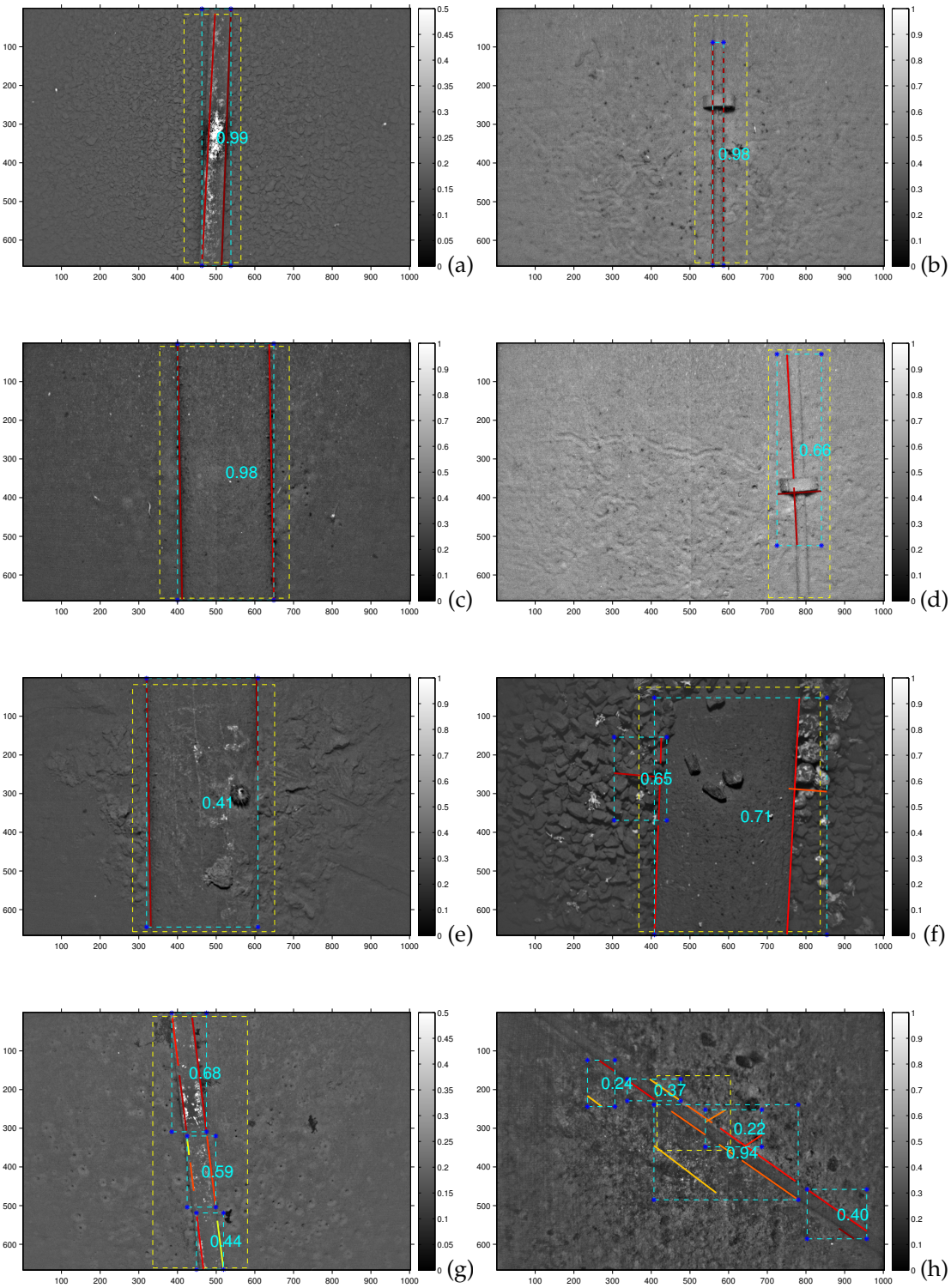
- Detektoren fant på det meste 75% av alle objekter markert i fasiten, men noe avvik var forventet. I forhold til det forventede antall deteksjoner (ved å ta hensyn til valgt modell og «umulige bilder») fant detektoren 90% av det estimerte antallet.
- Det oppstod relativt mange feildeteksjoner. Mange kan fjernes ved å sette en terskel for akseptabel objektkvalitet, men dette vil gå på bekostning av antall korrekte deteksjoner.
- Halvparten av fasitobjektene ble bare detektert med ett primitiv, eller ikke detektert i det hele tatt.

Oppsummering av eksempelbilder

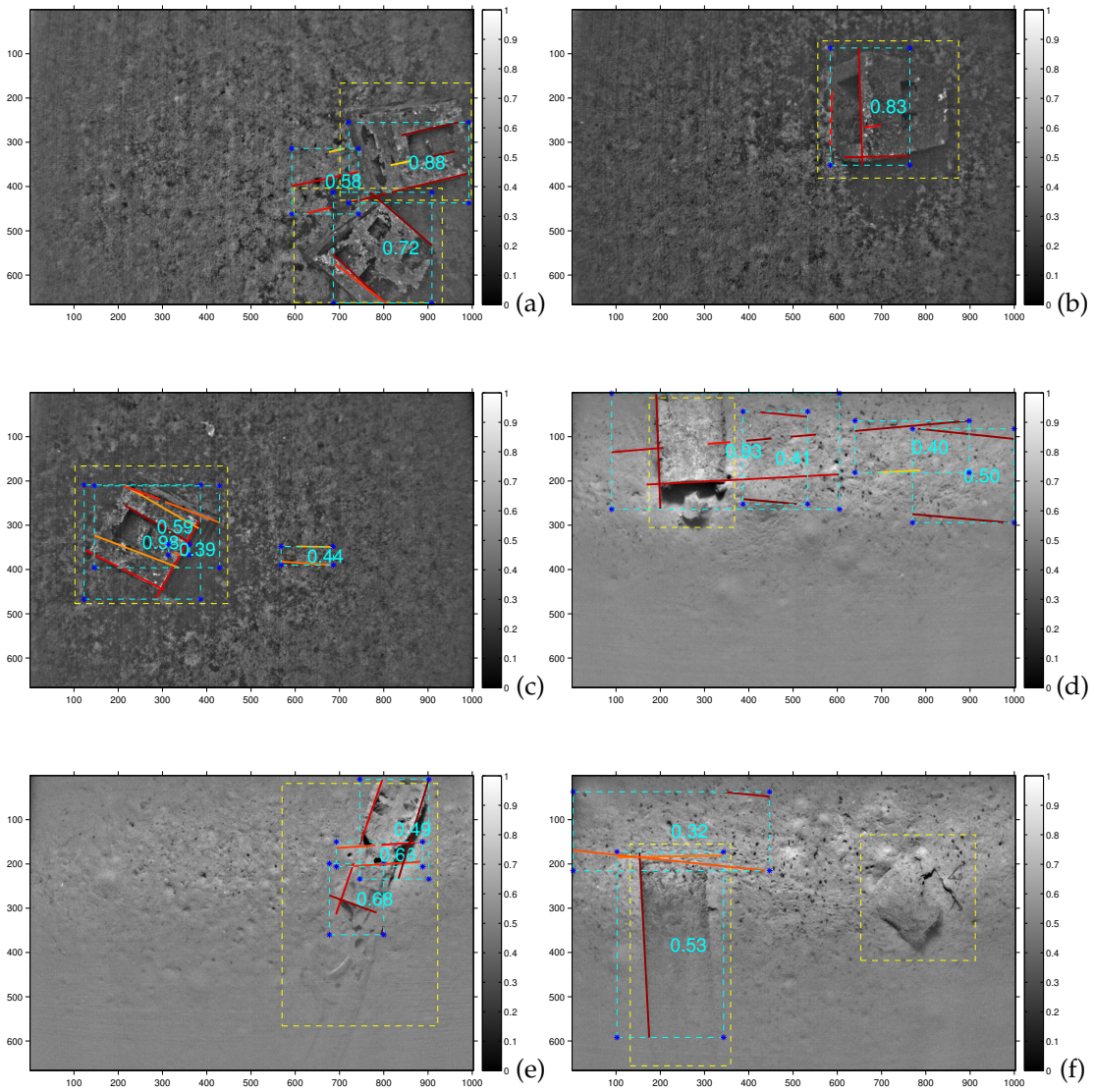
- På generell basis ser valgt modell ut til å være egnet for deteksjon av menneskeskapte objekter.
- Oljefat passer godt med modellen, og ser ut til å gi generelt gode treff.
- Flere gode deteksjoner av rør, men også noen rør som detekteres av flere spredte enkeltdeteksjoner. Det kommer også frem at rør ikke alltid detekteres av paralleller.
- Hovedårsaken til feildeteksjoner er at det trekkes ut linjesegmenter som ikke svarer til faktiske kanter i bildene.



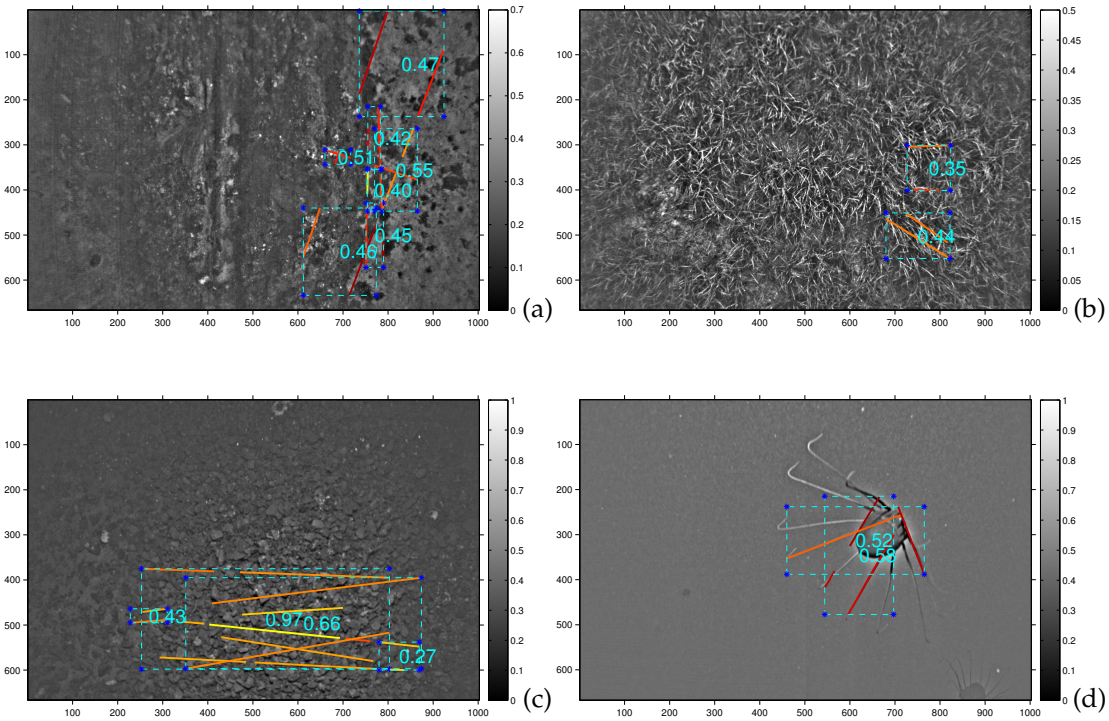
Figur 9.2: *Deteksjoner av enkeltobjekter. Noen av disse er svært relevante i forhold til egenskapene som detektoren leter etter, og gir høy konfidens (objektkvalitet). Bilde (a) til (c) viser oljefat. Bilde (d) viser en slags ramme som består flere like rektangler. Objekter er delvis gått i oppløsning. Bildene (e) og (f) viser en struktur som er koblet på et rør på havbunnen. Mange tydelige kanter og regelmessige geometriske former gir høy deteksjonskonfidens. Bilde (g) og (h) viser noe som ligner vrakgods.*



Figur 9.3: *Deteksjoner av rør. I både (a),(b) og (c) er det detektert parallell-primitiver som spenner over nesten hele bildet og som får god kvalitet. I (d) har det blitt deteksjon på grunn av et detektert tverrsegment, og i (e) får objektet lavere kvalitet på grunn av lengden av det korteste segmentet. I (f) til (h) har det skjedd flere deteksjoner, men på forskjellige steder av ett og samme objekt.*



Figur 9.4: *Mer eller mindre vellykkede deteksjoner av andre typer menneskeskapte objekter. Både (d) og (f) inneholder sylindriske objekter der det ikke lykkes å detektere parallell-primitiver.*



Figur 9.5: *Eksempler på feildeteksjoner på havbunn og naturlige objekter. Bilde (a) viser berggrunn med rette bruddflater som går langsetter bildet. Feildeteksjonene oppstår likevel i utkanten, på sandbunn med spredt vegetasjon. Sjøgress «Posidonia» er avbildet i (b), en antatt utfordrende havbunn. Bilde (c) viser stein og grus, med mange lange segmenter. I (d) har en manet blitt avbildet.*

Kapittel 10

Diskusjon og konklusjon

Dette kapittelet oppsummerer oppgaven og inneholder en diskusjon av valgene som ble tatt underveis i arbeidet med forslag til mulige forbedringer. Til slutt nevnes noen forslag til videre utvikling av detektoren.

10.1 Oppsummering

For å nå målet for oppgaven – *å gjøre automatisk deteksjon av menneskeskapte objekter i optiske undervannsbilder samlet inn av HUGIN AUV* – har det blitt utviklet et eget dataprogram som leser et undervannsbilde og gir ut en liste over eventuelle deteksjoner samt et mål på tiltro til hver deteksjon. Modellen som er valgt for å definere et objekt som menneskeskapt er å kreve at konturene (eller andre deler av objektet) danner rette linjer, og videre at det finnes minst to linjer som er enten parallelle eller ortogonale. For å vurdere ytelsen er detektorprogrammet testet på et relativt stort og variert sett av bilder.

I programmet utføres det en preprosessering i form av geometrisk korreksjon og bakgrunnsutjevning med homomorf filtrering på det innleste bildet. Deretter blir kanter segmentert med et fasekongruensfilter. Par av segmenterte kanter sammenlignes for å undersøke om linjesegmentene er parallelle eller står vinkelrett på hverandre. Slike hjørner og paralleller er selve grunnkomponentene i objektdeteksjonen, og er derfor omtalt som «primitiver». Dersom to eller flere primitiver deler en kant, vil disse slås sammen til en større gruppe som representerer én deteksjon. Jo flere kanter og primitiver som samles, jo sikrere blir detektoren på at gruppen svarer til et menneskeskapt objekt.

Resultatet er at detektoren oppnår en TPR på 0.75 som beste resultat. For denne verdien er laveste mulige FPR (*False Positive Rate*) 0.75. FPR kan reduseres betraktelig ved å stille strengere krav til deteksjonskvalitet (tiltro), men dette går relativt mye på bekostning av antall korrekte deteksjoner.

10.2 Diskusjon

Preprosessering Preprosesseringen som utføres i oppgaven vurderes generelt som tilfredsstillende. Hovedproblemet har vært en ujevn belysning som gir dårlig kontrast i kantene av bildet, men dette har latt seg løse med homomorf filtrering. Ulempen med metoden er at høypassfiltreringen forsterker støy og uinteressante kanter, som kan ha hatt negativ innvirkning på kantdeteksjon med fasekongruens.

Anisotrop diffusjon, medianfiltrering og morforlogisk lukking ble alle vurdert i forbindelse med støyfiltrering, men på grunn av fasekongruensfilterets innebygde støyfiltrering ble ikke disse tatt med i programmet. Når resultatene viser at det detekteres kanter på uventede steder i bildene, kan det tenkes at ekstra støyfiltrering likevel er aktuelt. I vurderingen av parametere for homomorf filtrering henvises det til figur 6.3(h) på side 33 som en illustrasjon på hvordan valg av en høyere knekkfrekvens kan resultere i et bilde tilnærmet uten informasjon. Dette er riktig, men det kunne vært interessant å bruke disse parametrene til å *lavpassfiltrere* bildet etter bakgrunnsutjevning for å fjerne høyfrekvent støy i bildet og dermed mange små kanter som detekteres med fasekongruens.

Med mer tid til rådighet ville det vært interessant å teste flere metoder i denne delen av oppgaven. Et alternativ som kunne vært vurdert for bakgrunnsutjevning er adaptiv histogramutjevning av typen *Contrast Limited Adaptive Histogram Equalization (CLAHE)*, som antas å være bedre egnet enn bottom-hat transform som ble testet i oppgaven.

Metode for segmentering av objekter To hovedtyper av metoder ble vurdert for segmentering av objekter (kanter): skyggedeteksjon ved terskling og noen varianter av kantdeteksjon. Det ble valgt å gå for kantdeteksjon, både på grunn av bildenes karakter og på grunn av vanskeligheter med valg av terskelverdier. I vurderingen av metodene var det stort fokus på støy og uønskede segmenterte objekter. Ved nærmere ettertanke kunne det vært lagt mindre vekt på dette og heller prøvd houghtransform og videre prosessering på både tersklet gradientmagnitude og tersklet skygge. Det har uansett vist seg å være behov for en bedre metode for å skille ut falske linjesegmenter.

Det finnes mange metoder for bildesegmentering, og et raskt litteratursøk avdekker flere oppsummeringsartikler som tar for seg de fleste kjente varianter. I en av disse artiklene[8] deles bildesegmenteringsmetoder i to kategorier: deteksjon av diskontinuitet (f.eks. kanter) og deteksjon av likhet (terskling og regiongroing). I de to kategoriene ligger mange undergrupper, men det var interessant å se at begge hovedkategorier faktisk har blitt prøvd i oppgaven.

Et alternativ som ikke ble testet, men som kunne være av interesse, er aktive konturer *(snakes)*. Slike metoder krever gjerne en ledet initialisering i et spesielt område av interesse, og en tanke er at *snakes* kunne vært brukt i tilfeller der bare ett linjesegment detekteres på objektet.

Modell for menneskeskapte objekter Å bruke rette vinkler eller parallelle linjer til deteksjon av menneskeskapte objekter vurderes som et fornuftig valg. I inspeksjonen

av testsettet ble bare 15 av 92 objekter (16%) vurdert som ikke detekterbare, og da var også bilder med dårlig kontrast medregnet. Konklusjonen er at mange av objektene i datasettet faktisk hadde en eller begge av disse egenskapene. Likevel er det verdt å merke seg at *perspektiv* kan avgjøre om to linjer ser parallelle ut eller ikke, og den konkrete grensen (målt i vinkler) for forskjell i vinkel spiller inn på resultatet. I oppgaven er denne satt til (90) $\pm 3^\circ$, som var for lite for et par av objektene i treningssettet.

En naturlig utvidelse av modellen vil være å lete etter andre geometriske former som sirkler og ellipser ved hjelp av houghtransform.

Utvikling av detektoren og testresultater Det aller meste av prosessen etter kantdeteksjon og radontransform består av egenutviklede metoder. Dette har vært en utfordrende og tidkrevende del av arbeidet. Metoder for deteksjon av linjesegmenter – fra toppdeteksjon i radonmatrisen til representasjon av radonlinjer (også med økt linjebredde) og uttrekking av segmenter basert på egenskaper i bildet – bør være kjente problemstillinger som kan ha løsninger beskrevet i litteraturen, mens grupperingen av linjesegmenter til større objekter og utviklingen av kvalitetsmål for deteksjoner i større grad må regnes som *ad hoc*[1]-metoder.

Selve prosessen med å sammenligne linjesegmentenes posisjon og orientering i forhold til hverandre vurderes som robust, selv om målet på kvalitet igjen blir mer *ad hoc*. Her kunne det vært brukt mye tid på å finne sofistikerte løsninger, men til slutt må det likevel velges et alternativ. Å bruke metoder inspirert av fuzzy logic til denne typen problemstilling har virket som et passende valg.

Tatt i betraktning modellen for menneskeskapte objekter og tilgjengelig datasett, vurderes programmets evne til å detektere menneskeskapte objekter som tilfredsstillende. Fordi nær halvparten av de faktiske objektene detekteres med bare ett primitiv, kreves det imidlertid at terskelen for tiltro til deteksjonene settes lavt for å oppnå en høy deteksjonsrate. Lav terskelverdi har vist seg å også medføre en høy feilrate.

Fra inspeksjonen av resultatene viser det seg at målet på deteksjonskvalitet ikke er nok til å skille ekte og falske deteksjoner som består av ett primitiv. Sett bort fra problemet med uventet mange falske linjesegmenter vil det fortsatt være behov for å skille deteksjoner på objekter og tilfeldige naturlige formasjoner. En løsning som foreslås er å se på pikselverdiene i arealet mellom segmentene, og bruke egenskaper som for eksempel homogenitet til å vurdere om området består av havbunn eller om segmentene avgrenser et objekt. Dette åpner for et annet alternativ, der man generelt ser på området mellom to segmenter som ligger nær hverandre, uten å stille krav til vinkelforskjell. I utgangspunktet *burde* enkeltdeteksjoner av rette linjer svare til en viss sannsynlighet for tilstedeværelse av et menneskeskapt objekt. Dette understøttes av testresultatet, som viser at enkelte rør faktisk ikke detekteres av parallelle linjer, men av et enkeltsegment og en (i blant tilfeldig) ortogonal.

I stedet for den «fuzzy» metoden som er brukt for deteksjon av ortogonale linjer i oppgaven, kunne det være interessant å prøve hjørnedeteksjon ved hjelp generalisert

[1] Fra latin: «til dette», underforstått «til dette formål».

houghtransform [9].

Observasjoner i etterkant Gjennom inspeksjonen av testresultatene og diskusjonen forøvrig har det gjentatte ganger blitt nevnt et problem med uventet mange linjesegmenter i områder uten faktiske kanter. Dette har vært særlig fremtredende i bilder som generelt inneholder få og svake kanter. Problemet har etter all sannsynlighet stor sammenheng med at det er utført en *ekstra* normalisering av fasekongruens. Dette er noe som henger igjen fra eksperimentering med gradientmagnitude, som i utgangspunktet er et ubegrenset mål (kan anta svært høye verdier). Fasekongruens derimot, er som beskrevet i avsnitt 7.4 allerede et normalisert mål med verdier mellom 0 og 1. Ved å normalisere en ekstra gang vil fasekongruensmagnituden i bilder som nesten ikke har kanter blåses opp og stille på lik linje med veldefinerte objekter i bilder med god kontrast. Problemet kan skyldes at utviklingen har foregått på et relativt lite treningssett, og at det dermed oppstod noen uforutsette forhold ved testing på et vesentlig større testsett. Å fjerne funksjonskallet innebærer å slette én linje fra programkoden, men oppdagelsen er gjort på et såpass sent tidspunkt at det er besluttet å ikke gjøre noen endringer, hverken i programmet eller i rapporten. Feilrettingen ville gitt nye forutsetninger for eksperimenter og metoder i den siste delen av arbeidet, og dessverre medført for mye arbeid innenfor oppsatt tidsfrist. Dette har vært en bitter, men verdifull erfaring.

Målet for segmentkvalitet har også blitt revurdert i ettertid, da det viste seg å være lite egnet til å skille ekte og falske kantdeteksjoner. Et kort eksperiment ble gjort for å teste to andre egenskaper: sum av fasekongruens langs et linjesegment, og avvik fra forventet retning regnet ut på en alternativ måte. Studien ble gjort som en kuriositet etter at alle resultater var regnet ut og store deler av rapporten var skrevet, så det ble besluttet å ikke endre kvalitetsmålet selv om sum av fasekongruens viste seg å være en bedre egenskap. Det presiseres at studien ble gjort *før* det ovennevnte problem ble oppdaget.

10.3 Videre arbeid

I dette avsnittet er det samlet noen forslag og ideer til videre utvikling, som kommer i tillegg til det som allerede er nevnt i diskusjonen.

Gjøre bruk av enkeltsegmenter Det vil være nyttig å kunne lete etter objekter basert på bare én detektert kant. På grunn av hjelpebelysningen er det flere objekter der kantene hovedsaklig er synlige på skyggesiden av objektet.

Sekvenser av bilder Siden kameraet på HUGIN AUV kan ta bilder med en rate på opptil fire bilder per sekund, kan det være interessant å bruke sekvenser av bilder for å detektere objekter. Ved rørinspeksjon kan dette utnyttes for å predikere hvor røret vil befinne seg i neste bilde. En annen tenkt anvendelse kan være simulert stereosyn for å danne 3D-modeller av scenen.

Multistråle-ekkolodd Å bruke sensordata fra multistråle-ekkolodd til å lokalisere objekter og definere områder av interesse i bildeflaten vil antagelig kunne øke ytelsen til detektoren betraktelig.

Vedlegg

Vedlegg A

Programkode

Programkode A.1: *Homomorf filtrering*

```
%% Homomorphic filtering
% [imOut] = HMMF(imIn, boost, order, cutOff)
% imIn   : Image to filter
% boost  : Scaling of high pass filter [0 1]
% order  : Order of Butterworth filter
% cutOff : Cut off frequency (normalized)
% Typical values: HMMF(im, 0, 1, 0.01)
function [imOut, hp] = hmmf(imIn, boost, order, cutOff)
    [ny, ~] = size(imIn);
    imIn = double(imIn - min(imIn(:)));
    tmp = log(imIn + 1);                          % make log transform
    tmp = [tmp; tmp(end:-1:1,:)];                 % mirroring to avoid discont
    lp = lpbwfilter(size(tmp), cutOff, order);    % create lowpass filter
    hp = (1 - boost)*(1 - lp) + boost;            % create highpass filter
    tmp = real(ifft2(fft2(tmp).*hp));             % filter in freq. domain
    tmp = tmp(1:ny, :);                           % remove mirroring
    tmp = exp(tmp);                               % inverse log transform
    % centre the color map at histogram peak
    [low, ~, high] = intensityLims(tmp);
    imOut = min(max(tmp-low,0)/(high-low),1.0);
    imOut(imOut == 4095) = 1.0;
end
```

Figurer

Tabeller

Akronymer

AUV	Autonomous Underwater Vehicle
CLAHE	Contrast Limited Adaptive Histogram Equalization
FFI	Forsvarets forskningsinstitutt
GLCM	gray-level co-occurence matrix
KM	Kongsberg Maritime
LED	lysdiode
MBE	Multi Beam Echo Sounder
NEO	Norsk Elektro Optikk
NUI	Norsk Undervannsintervensjon
ROV	Remotely Operated Vehicle
SAS	syntetisk aperture sonar
SSS	side-scan sonar
UUV	Untethered Underwater Vehicle

Bibliografi

[1] Victor Adam. *William Thompson- 100 years of underwater photography?* Sep. 1993. URL: http://www.bsoup.org/Articles/William_Thompson.php (sjekket 07.07.2014).

[2] Fritz Albregtsen. «Non-contextual thresholding, Adaptive thresholding». University lecture. 13. feb. 2008. URL: http://heim.ifi.uio.no/~inf5300/2008/thresholding-13022008.pdf (sjekket 08.04.2014).

[3] Fritz Albregtsen. «Segmentering ved terskling». University lecture. 7. mai 2013. URL: http://www.uio.no/studier/emner/matnat/ifi/INF2310/v13/undervisningsmateriale/forelesningsnotater/inf2310-2013-13-segmentering-4fpp.pdf (sjekket 28.04.2014).

[4] Fritz Albregtsen. «Texture». University lecture. 7. sep. 2011. URL: http://www.uio.no/studier/emner/matnat/ifi/INF4300/h11/undervisningsmateriale/INF4300-2011-f02-texture.pdf (sjekket 02.05.2014).

[5] Andreas Arnold-Bos, Jean-Philippe Malkasse og Gilles Kervern. «Towards a model-free denoising of underwater optical images». I: *Oceans 2005-Europe*. Bd. 1. IEEE. 2005, s. 527–532.

[6] D Richard Blidberg. «The development of autonomous underwater vehicles (auvs); a brief summary». I: URL: http://ausi.org/publications/ICRA_01paper.pdf (sjekket 07.07.2014).

[7] J. E. Bresenham. «Algorithm for computer control of a digital plotter». I: *IBM Systems Journal* 4.1 (1965), s. 25–30.

[8] Rajeshwar Dass og Swapna Devi. «Image Segmentation Techniques». I: *The International Journal of Electronics & Communication Technology (IJECT)* 3.1 (jan. 2012), s. 66–70.

[9] E. R. Davies. «Application of the generalised hough transform to corner detection». I: *IEE Proceedings E (Computers and Digital Techniques)* 135.1 (1988), s. 49–54.

[10] Stanley R. Deans. «Hough Transform from the Radon Transform». I: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* PAMI-3.2 (mar. 1981), s. 185–188.

[11] New Oxford American Dictionary. *Robot*. Red. av Angus Stevenson og Christine A. Lindberg. 2014.

[12] Richard O Duda, Peter E Hart og David G Stork. *Pattern classification*. 2. utg. John Wiley & Sons, 2001.

[13] R. Garcia, T. Nicosevici og X. Cufi. «On the way to solve lighting problems in underwater imaging». I: *OCEANS '02 MTS/IEEE*. Bd. 2. 2002, 1018–1024 vol.2. DOI: 10.1109/OCEANS.2002.1192107.

[14] Thomas Giddings, Cetin Savkli og Joseph Shirron. *Image Enhancement and Automated Target Recognition Techniques for Underwater Electro-Optic Imagery*. Tekn. rapp. DTIC Document, 2006.

[15] Rafael C Gonzalez og Richard E Woods. *Digital imaging processing*. 3. utg. Prentice Hall, 2008.

[16] Per Espen Hagen mfl. «Making AUVs truly autonomous». I: *Underwater Vehicles*. Red. av Alexander V. Inzartsev. In-Tech Education og Publishing, Vienna, Austria, 2008. Kap. 8, s. 129–152.

[17] Jan O. Hallset. «Simple vision tracking of pipelines for an autonomous underwater vehicle». I: *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. IEEE. 1991, s. 2767–2772.

[18] Roy Edgar Hansen. «Introduction to Synthetic Aperture Sonar». I: *Sonar Systems*. Red. av Nikolai Kolev. InTech, 2011, s. 3–28. ISBN: 978-953-307-345-3.

[19] Janne Heikkilä og Olli Silvén. «A four-step camera calibration procedure with implicit image correction». I: *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*. IEEE. 1997, s. 1106–1112.

[20] Kashif Iqbal mfl. «Underwater Image Enhancement Using an Integrated Colour Model.» I: *IAENG International Journal of Computer Science* 34.2 (2007).

[21] Jason C Isaacs og Ross Goroshin. «Automated cable tracking in sonar imagery». I: *OCEANS 2010*. IEEE. 2010, s. 1–7.

[22] J.S. Jaffe. «A historical perspective on underwater optical imaging». I: *OCEANS - Bergen, 2013 MTS/IEEE*. Jun. 2013, s. 1–3. DOI: 10.1109/OCEANS-Bergen.2013.6608121.

[23] Michael Kabatek, Mahmood R Azimi-Sadjadi og J Derek Tucker. *An underwater target detection system for electro-optical imagery data*. IEEE, 2009.

[24] Andreas Kleppe. «Filtrering i bildedomenet - II». University lecture. 3. mar. 2014. URL: http://www.uio.no/studier/emner/matnat/ifi/INF2310/v14/undervisningsmateriale/forelesningsnotater/inf2310-2014-07-filtrering-ii-1fpp.pdf (sjekket 25.04.2014).

[25] Donna M Kocak mfl. «A focus on recent developments and trends in underwater imaging». I: *Marine Technology Society Journal* 42.1 (2008), s. 52.

[26] Peter Kovesi. *Anisotropic diffusion*. URL: http://www.csse.uwa.edu.au/~pk/research/matlabfns/#anisodiff (sjekket 12.02.2014).

[27] Peter Kovesi. «Image features from phase congruency». I: *Videre: Journal of computer vision research* 1.3 (1999), s. 1–26.

[28] Peter Kovesi. *Phase Based Feature Detection and Phase Congruency*. URL: http://www.csse.uwa.edu.au/~pk/research/matlabfns/#phasecong (sjekket 12.02.2014).

[29] Peter Kovesi. «Phase congruency detects corners and edges». I: *The australian pattern recognition society conference: DICTA 2003*. 2003.

[30] Thomas R Krogstad mfl. «Autonomous object identification in mine countermeasure missions». I: *OCEANS 2011*. IEEE. 2011, s. 1–9.

[31] Isabelle Leonard, A Arnold-Bos og Ayman Alfalou. «Interest of correlation-based automatic target recognition in underwater optical images: theoretical justification and first results». I: *SPIE Defense, Security, and Sensing*. International Society for Optics og Photonics. 2010, 76780O–76780O.

[32] Nicolas Mandelert og Andreas Arnold-Bos. «Sonar and Video Perception for an Autonomous Mine Disposal Vehicle». I: *Proc. Conf. Undersea Defence and Technology*. 2008.

[33] Øivind Midtgaard mfl. «Imaging sensors for AUVs in military operations». I: *Proc. NATO RTO SET* 169 (2011).

[34] Pietro Perona og Jitendra Malik. «Scale-space and edge detection using anisotropic diffusion». I: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 12.7 (1990), s. 629–639.

[35] Stanford Encyclopedia of Philosophy. *Fuzzy Logic*. URL: http://plato.stanford.edu/entries/logic-fuzzy/ (sjekket 17.06.2014).

[36] Raimondo Schettini og Silvia Corchs. «Underwater image processing: state of the art of restoration and image enhancement methods». I: *EURASIP Journal on Advances in Signal Processing* 2010 (2010).

[37] Ragnar Smestad. «Correction of TileCam underwater images». FFI-rapport 2013/02124. 14. okt. 2013.

[38] Spandan Tiwari. *Matlab central: Steve on Image Processing, Homomorphic filtering*. URL: http://blogs.mathworks.com/steve/2013/06/25/homomorphic-filtering-part-1/ (sjekket 04.12.2013).

[39] Øivind Due Trier og Anil K. Jain. «Goal-directed evaluation of binarization methods». I: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 17.12 (1995), s. 1191–1201.

[40] Roger Tsai. «A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses». I: *Robotics and Automation, IEEE Journal of* 3.4 (1987), s. 323–344.

[41] Gergely Vass og Tamás Perlaki. «Applying and removing lens distortion in post production». I: *Proceedings of the 2nd Hungarian Conference on Computer Graphics and Geometry*. Citeseer. 2003, s. 9–16.

[42] Aaron Wetzler. *Bresenham optimized for Matlab*. URL: http://www.mathworks.com/matlabcentral/fileexchange/28190-bresenham-optimized-for-matlab (sjekket 10.02.2014).

[43] Wikipedia. *Anisotropic diffusion*. URL: http://en.wikipedia.org/wiki/Anisotropic_diffusion (sjekket 12.05.2013).

[44] Wikipedia. *Beer–Lambert law*. URL: http://en.wikipedia.org/wiki/Beer-Lambert_law (sjekket 18.10.2013).

[45] Wikipedia. *Computer stereo vision*. URL: http://en.wikipedia.org/wiki/Computer_stereo_vision (sjekket 24.01.2014).

[46] Wikipedia. *Drum (container)*. URL: http://en.wikipedia.org/wiki/Drum_(container) (sjekket 20.09.2013).

[47] Wikipedia. *Sobel operator*. URL: http://en.wikipedia.org/wiki/Sobel_operator (sjekket 06.01.2014).

[48] S.D. Yanowitz og A.M. Bruckstein. «A new method for image segmentation». I: *Computer Vision, Graphics, and Image Processing* 46.1 (1989), s. 82–95. ISSN: 0734-189X. DOI: http://dx.doi.org/10.1016/S0734-189X(89)80017-9. URL: http://www.sciencedirect.com/science/article/pii/S0734189X89800179.